Universidad
Carlos III de Madrid

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

PROYECTO FIN DE CARRERA

Hmi_wiimote: La integración del mando inalámbrico Wii Remote dentro de la arquitectura software YARP

Autor: Rubén Rodrigo Valero
Tutor**:** Alberto Jardón Huete
Co-Tutor: Juan Carlos González Víctores

Titulación**:** I.T.I. Electrónica Industrial

LEGANÉS, MADRID
NOVIEMBRE 2010

TÍTULO: Hmi_wiimote: la integración del mando inalámbrico Wii Remote dentro de la arquitectura software YARP

AUTOR: RUBÉN RODRIGO VALERO

TUTOR: ALBERTO JARDÓN HUETE

CO-TUTOR: JUAN CARLOS GONZÁLEZ VÍCTORES

La defensa del presente Proyecto Fin de Carrera se realizó el día 24 de Noviembre de 2010; siendo calificada por el siguiente tribunal:

PRESIDENTE: *Santiago Martínez de la Casa*

SECRETARIO: *Miguel González-Fierro*

VOCAL: *Agapito Ledezma Espino*

Habiendo obtenido la siguiente calificación:

CALIFICACIÓN:

Presidente **Secretario** **Vocal**

AGRADECIMIENTOS

En primer lugar me gustaría agradecer a Alberto Jardón la oportunidad que me brindó para desarrollar este proyecto que me ha servido para cambiar mi forma de enfrentarme a nuevos retos y ha supuesto una experiencia muy positiva teniendo en cuenta mi escasa experiencia en este ámbito.

En segundo lugar, deseo agradecer a Juan Carlos González Víctores, más conocido como Juan G. Victores, su grandísimo apoyo en todo momento a pesar de tener muchas obligaciones que cumplir. También por su paciencia a la hora de explicarme conceptos no muy fáciles de comprender considerando mis pequeños conocimientos de programación y robótica, y lo más importante de todo, agradecerle la forma de trabajar que me ha enseñado, siempre eficiente y dinámica.

Cómo no, agradecer a mis padres, a mi hermana y a toda la familia la comprensión y el apoyo incondicional que me han brindado a lo largo de todos estos años y especialmente en los momentos duros de la carrera así como a la hora de tomar decisiones importantes.

Por último agradecer a mis amigos, tanto compañeros de la carrera, que ya lo son de por vida, como a los más antiguos, su presencia permanente en cualquier situación y su buen humor, que han conseguido que lleve un pedazo de su personalidad en mi.

A todos ellos, gracias de corazón.

RESUMEN

El grupo Robotics Lab de la Universidad Carlos III de Madrid, desarrolló allá por 2004 un primer prototipo de ASIBOT, un robot cuya finalidad era la ayuda y asistencia a personas discapacitadas y de la tercera edad.

En 2008, gracias al trabajo del personal del departamento, se mejoraron las características del prototipo y se consiguió eliminar problemas encontrados en las fases de pruebas con usuarios reales de la anterior versión modificando ciertos componentes para superar las limitaciones del sistema de comunicaciones interno.

Este proyecto fin de carrera forma parte de dicha evolución de manera que el objetivo principal es crear un software englobado en esta arquitectura llamada RCGv03 (Robot Component Guidelines v0.3) usada en la nueva evolución y cuya finalidad sea utilizar el periférico *Wii Remote*® de la conocida videoconsola *Wii*® de *NINTENDO*® para controlar el brazo articulado y realizar sus tareas.

Al estar trabajando en un entorno donde se respetan las pautas que se vienen siguiendo desde la creación de este tipo de proyectos (RCGv03), este módulo es reutilizable para controlar cualquier dispositivo con las mismas características y es compatible con cualquier otro módulo de tal naturaleza como se demuestra al final del documento al utilizar el software creado por el compañero de trabajo Daniel García Sánchez, que consiste en una visualización de los datos enviados por un sensor mediante la herramienta “YARP view” en una pequeña ventana con varias líneas que representan las distintas variables del sensor además de ser capaz de mover los cinco grados de libertad del citado ASIBOT mediante la librería YARP y las conexiones pertinentes. Inicialmente el módulo de Daniel García Sánchez se creó para ser utilizado con un sensor de tipo MTi Xsens, pero gracias a esta estandarización de software es posible compatibilizarlo con el mando *Wii Remote* y visualizar sin ningún problema las cinco variables que porciona.

ABSTRACT

Back in 2004, the Robotics Lab at the University Carlos III of Madrid developed a first prototype of ASIBOT, a robot designed to help and assist the physically handicapped and the elderly. In 2008, thanks to the work of the department staff, improvements were made in this prototype and problems in the first version, which had been found during the testing stages with real-life users, were eliminated. This was achieved by modifying certain components in order to circumvent limitations in the internal communication system.

This final undergraduate project comes to form part of this process, in that its main objective is to design software to be included within the software architecture called RCGv03 (Robot Component Guidelines v0.3), which is used in the later stages of this process. The purpose of this new software is to use the *Wii Remote* from the well-known *Wii* video game console by *NINTENDO* to control the robot´s articulated arm and to perform its tasks.

Having been created in an environment which continues to adhere to the original guidelines established for these types of projects (RCGv03), this module can be used to control any device with the same specifications, and is compatible with any other module which shares the same architecture. This is demonstrated at the end of the document when using software designed by my colleague, Daniel García Sánchez. This software uses a tool called "YARP view" to display the data sent by a sensor in a small window with several colored lines, each of which represents a different variable provided by the sensor. The software is also capable of moving the five degrees of freedom of the aforementioned ASIBOT using the YARP library and the appropriate connections. Originally, the module by Daniel García Sánchez was meant to be used with a MTi Xsens sensor. However, thanks to this standardization in software, it can also be made compatible with the *Wii Remote* and the five variables it provides can be viewed without difficulty.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

Dado que el envejecimiento de la población es un hecho, la sensibilización por parte de la comunidad científica e investigadora se hace más fuerte día a día prestándose en favor de este sector de la población que a menudo padece enfermedades degenerativas y que requieren de un cuidado constante y determinado.

1.1. MOTIVACIÓN – ENTORNO WII Y DESARROLLO DE PROYECTOS ASIBOT

Las nuevas tecnologías están cada vez más presentes hoy en día, mejorando la calidad de vida y haciendo que las actividades cotidianas sean más sencillas. Las personas con limitaciones funcionales ya sean de movilidad, de tipo cognitivo o bien debido a problemas relacionados con el envejecimiento o cualquier otra enfermedad, deben adaptarse a las nuevas tecnologías, así como éstas a las nuevas necesidades de la sociedad, lo que conlleva serias dificultades en su desarrollo y utilización. La adaptación a las mismas no sólo es necesaria para realizar su plena integración en la sociedad, sino que también debe ser una herramienta para mejorar su calidad de vida.

El *Wiimote Plus* tiene diferentes características que lo convierten en un dispositivo interesante para desarrollar aplicaciones para todos los sectores de la población:

- Utilización sencilla y control intuitivo que ha generado una gran aceptación entre todos los públicos.
- Emplea un estándar de comunicaciones, permitiendo que cualquier dispositivo que haga uso del mismo pueda tener acceso a la información que envía el *Wiimote*.
- Gran cantidad de gente involucrada en su ingeniería inversa, incluyendo librerías de acceso gratuito para interactuar con el controlador.
- Posee gran cantidad de sensores integrados en un pequeño espacio, a un precio muy reducido (en torno a 40 euros).

La videoconsola *Wii* ha sido diseñada con el valor añadido del entretenimiento para todos, por lo que no es sorprendente que el *Wiimote* se esté convirtiendo en interfaz de aplicaciones destinadas a mejorar la calidad de vida de ancianos o personas enfermas o con discapacidad.

Este controlador puede aportar al mencionado sector de la población la posibilidad de disfrutar de juegos y aplicaciones que de otro modo estarían fuera de su alcance.

También se plantea como herramienta para terapias de rehabilitación y en el control de pacientes. En general, la comunidad investigadora ha propuesto este mando como solución en diferentes proyectos, utilizando en parte todo su potencial.

En definitiva, esta nueva tecnología puede ayudar a las personas a desarrollar actividades que de otro modo sería imposible llevar a cabo.

En la Universidad Carlos III de Madrid, hace años que se está desarrollando un robot asistencial, conocido como ASIBOT [1], que consiste en un brazo capaz de acoplarse a infinidad de lugares gracias a unas fijaciones metálicas. En la figura 1.1 puede observarse cómo se encuentra acoplado a una de estas fijaciones metálicas, que a su vez se encuentra sujeto a una silla de ruedas.

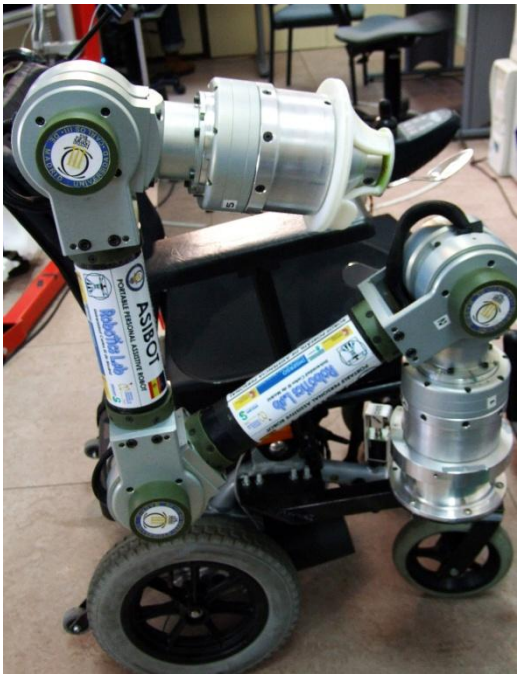


Figura 1.1: ASIBOT acoplado a silla de ruedas

El robot es capaz de realizar tareas de gran ayuda a discapacitados como beber agua, lavarse los dientes, la cara, etc. Una de las líneas de investigación más importantes para el robot se centra en el desarrollo de interfaces multimodales para su control y mando.

1.2. OBJETIVOS DEL PROYECTO

En este proyecto se desarrolla un caso particular de las infinitas aplicaciones de los periféricos de la videoconsola *Wii*®: la adaptación de su controlador de videojuegos *Wiimote Plus*®, fabricado y distribuido por *Nintendo*®. Se va a desarrollar un nuevo módulo que servirá para comunicar los datos enviados por el *Wiimote* con un ordenador remoto mediante el uso de librerías y puertos YARP, que tendrá una estructura compatible con otros módulos de software englobados en proyecto ASIBOT y cuya funcionalidad se desarrollará con detenimiento en los capítulos siguientes.

Para hacer posible este objetivo principal es necesario concluir varios apartados previos:

- Correcta instalación en el equipo de las aplicaciones y librerías necesarias para el funcionamiento del interfaz de comunicaciones.

- Modificación del código en función de las necesidades de comunicación entre el mando inalámbrico y el equipo.

- Estandarización del módulo respecto a la arquitectura creada previamente para su compatibilización con otros dispositivos.

- Comprobación de los resultados obtenidos y experimentación con dispositivos compatibles con la arquitectura del módulo.

1.3. ESTRUCTURA DEL DOCUMENTO

Este documento consiste en seis secciones o capítulos bien diferenciados en los que se expone el proyecto que se ha llevado a cabo visto desde todas sus dimensiones como son hardware, software, contexto en el que se trabaja, resultados previos, etc.

En esta primera sección se introduce de manera breve la tecnología de la que se dispone y sobre la que se ha trabajado para alcanzar los objetivos principales expuestos en la sección 1.2, aparte de las características principales que presenta el dispositivo utilizado y las razones por las que se considera muy útil en el sector de población con el que se está trabajando, principalmente personas de la tercera edad y discapacitados.

La segunda sección consiste en una presentación de los diversos dispositivos y periféricos con los que cuenta *Nintendo* y que están resultando muy atractivos a nivel usuario pero a su vez están sirviendo de base para investigaciones médicas y tecnológicas que pretenden mejorar la calidad de vida del sector de la población con el que se está tratando y que en muchos casos lo han conseguido como la conocida “Wii terapia” que se ha extendido por hospitales y asociaciones de todo el mundo.

La tercera sección engloba toda la tecnología en la cual consiste este proyecto del módulo Hmi_wiimote a nivel hardware y software.La cuarta sección consiste en el análisis detallado de todas las modificaciones del código que han sido necesarias para la creación del módulo así como los problemas encontrados a lo largo del proceso y las distintas pruebas realizadas.

La quinta sección es el resultado obtenido y presentado de dos formas distintas, una en modo numérico y otra utilizando un interfaz gráfico. También se da una visión de la versatilidad de este tipo de módulos con su aplicación directa al robot ASIBOT y su capacidad para moverlo con cinco grados de libertad.

Por último, en la sexta sección se ofrece una conclusión del proyecto en general y se presentan ideas de futuras ampliaciones y dispositivos que pueden resultar útiles en este contexto en el que se ha trabajado.

CAPÍTULO 2. ESTADO DEL ARTE

En base a la tecnología existente en la actualidad se va a desarrollar un proyecto que utiliza el software propietario y el controlador remoto original de *Nintendo* para adaptar dicho software a una serie de estándares que lo compatibilizarán con multitud de aplicaciones.

2.1. EL EQUIPO WII DE NINTENDO Y SUS JUEGOS COMERCIALES

El mercado de los videojuegos, al igual que la informática y todas las nuevas tecnologías, está siempre en una constante evolución. Las compañías lanzan nuevos productos cada cierto tiempo y es habitual que cuando compramos un ordenador exista ya alguno mejor nada más salir de la tienda. El mundo de las videoconsolas no es ajeno a este caminar de la industria y las consolas tienen una vida de unos cinco años, tras los cuales un modelo es sustituido por otro más novedoso, potente y con todos los avances que han ido apareciendo durante ese tiempo. Desde que en 1983 surgió la primera consola de sobremesa de *Nintendo*, la Nintendo Entertaiment System o *NES*®, la evolución ha sido constante, con múltiples innovaciones que esta compañía nipona ha añadido al avance de la industria. Tras los 8 bits de *NES* llegaron los 16 de *Super NES*, con una gran calidad gráfica y colorido mejoradas. Años después llegó *Nintendo 64*®, que innovó de sobremanera en los juegos tridimensionales, sentando las bases de la jugabilidad en juegos 3D. La representante hasta ahora y durante los últimos años de *Nintendo* en el mercado mundial de videoconsolas ha sido *Gamecube*®, la cual ha ofrecido el renacer de viejas sagas junto con nuevas sagas que se añaden a la lista de creaciones de la Gran N con un colorido y jugabilidad únicos.

Fue hace ya unos años cuando *Nintendo* anunció su nuevo proyecto para suceder a *Gamecube*. Bajo el nombre clave "Revolution" se proponía todo un cambio en las bases de las videoconsolas tal y como las conocíamos hasta ahora, algo que *Nintendo* prometía y ya estaba mostrando con su portátil *Nintendo DS*®, aunque por aquel entonces no se sabía mucho. El secretismo que podemos encontrar alrededor de una nueva consola antes de su lanzamiento siempre es inmenso, pero con Revolution éste se vió incrementado en gran medida, no obstante, no se dieron detalles de la consola hasta un año antes de su lanzamiento, sin mostrar imágenes de los juegos hasta unos pocos meses antes.

Figura 2.1: Videoconsola *Nintendo Wii*. Imagen Google

"Revolution" no era más que un nombre temporal de aquella máquina que fue mostrada en el E3 (la expo de electrónica y entretenimiento) de 2005, sin mostrar imágenes de juegos o el propio mando de la consola, clave del sistema de juego. Fue el 27 de abril de 2006 cuando el mundo conoció el original nombre de la nueva generación de *Nintendo*, *Wii*, un nombre atípico para una consola de juegos que no significa nada, es corto y fácil de recordar; un nombre que originalmente no gustó demasiado, pero que con el paso del tiempo ha acabado en boca de todo el mundo, gente que incluso no ha jugado nunca a un videojuego.

¿Pero que es *Wii*? ¿Por qué está revolucionando el mundo de los videojuegos? La respuesta es sencilla y requiere conocer la forma en que han evolucionado los juegos a lo largo de los años. Inicialmente el mundo de los videojuegos contaba con muy poco potencial y los gráficos y todo tipo de efectos multimedia a los que estamos acostumbrados hoy en día eran imposibles de imaginar; eran tiempos donde la evolución sólo podía venir dada por nuevos géneros de juego o nuevas forma de representaciones visuales a base de "sprites" sencillos.

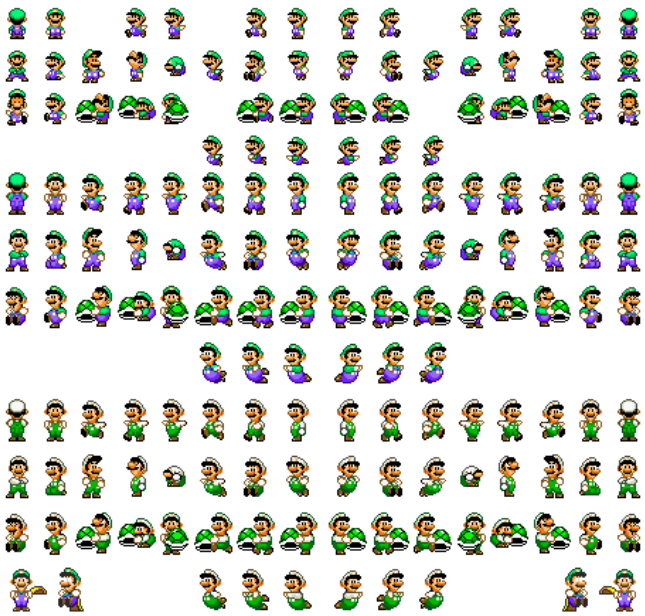


Figura 2.2: “Sprites” de Luigi, personaje creado por *Nintendo*. Imagen Google

Más tarde llegó la primera gran evolución en el mundo de los videojuegos desde que aparecieron los primeros gráficos (antes este tipo de juegos estaban basados en texto), los gráficos tridimensionales. Todas las evoluciones antes y después del uso de gráficos tridimensionales han sido únicamente aumentos de potencia, mejorar la calidad visual con más colores, efectos, número de polígonos, escenarios mayores y detallados… pero nada más. Es por esto por lo que *Wii* es una autentica evolución en los videojuegos, ya que cambia por completo el concepto de interactuar con ellos.

Figura 2.3: Familia jugando con la videoconsola *Wii*

Hasta ahora un videojuego se controlaba con un mando o teclado, a base de pulsar botones y realizar combinaciones que en muchos casos eran muy complejas, algo que suponía un gran aprendizaje para poder jugar con soltura. *Wii* rompe con esta idea y eso lo vemos claramente en su curioso mando de control, con un aspecto similar a los mandos de televisión (alargado y vertical), con pocos botones y que se maneja con una sola mano (aunque podemos conectarle el "nunchaku", una extensión conectada por un cable al mando principal y que añade dos botones más además de un joystick analógico).

2.2. EL MANDO INALÁMBRICO

La clave de *Wii* es su sistema de juego basado en el reconocimiento de movimiento. El mando de *Wii* [2] utiliza tecnología giroscópica que permite detectar el movimiento que realizamos con el mando por completo, además de la velocidad con que lo movemos. El mando, de forma alargada, es de un tamaño muy reducido y cuenta con ocho botones además de una cruceta, aunque durante los juegos el uso se limita casi por completo a dos de ellos, siendo los otros para realizar funciones concretas, incluso uno de ellos se utiliza en exclusiva para encender y apagar la consola a distancia. Este mando hace el papel de raqueta, palo de golf, espada, pistola, batuta… de forma que el control de estos elementos será totalmente intuitivo, moviendo el mando como si tuviésemos cualquiera de estos elementos en la mano.

El mando es totalmente inalámbrico y permite apuntar a la pantalla en ciertos títulos, al igual que ciertos controladores con forma de pistola. La alimentación del mando se realiza mediante dos pilas de las que se indica la carga mediante las luces que dispone el mando en la parte inferior, las cuales sirven también para indicar nuestro número de jugador. Ni la detección del movimiento, ni el hecho de ser un mando inalámbrico son problemas para que este pequeño controlador incluya vibración, y no sólo eso, sino que también cuenta con un pequeño altavoz que puede reproducir algunos sonidos o frases en momentos claves del juego para conseguir una involucración mayor por parte del jugador.

Aunque este mando es el corazón de *Wii*, no es la única forma de control en los juegos, ya que al mando principal *Wii Remote* podemos conectar una extensión conocida como "nunchaku", la cual va unida por un cable, de forma que con una mano manejamos el mando de *Wii* y con la otra el nunchaku. El nunchaku cuenta con dos gatillos y un joystick analógico, aunque también permite detectar cierto movimiento, en concreto movimientos rápidos que realicemos con él. Aun así algunos juegos pueden ser controlados con el mando clásico, un controlador con aspecto clásico que consiste en dos joysticks analógicos, gatillos, cruceta y cuatro botones.

Figura 2.4: Mando Wii Remote Plus

2.3. RESULTADOS PREVIOS CON DISPOSITIVOS DE NINTENDO

Algunos de los males más frecuentes en las sociedades avanzadas son el sedentarismo, el estrés, la falta de tiempo para programar dietas saludables y muchos otros más que provocan un deterioro de nuestro cuerpo cada vez más avanzado y un aceleramiento del envejecimiento muscular, de forma que Shigueru Miyamoto, el "padre" de los videojuegos, tuvo la gran idea de crear un software capaz de reunir a la familia delante del televisor y darles grandes dosis de entretenimiento con el pretexto de coger el hábito del ejercicio para tonificar el cuerpo.

Gracias a la gran variedad de juegos disponibles para la *Nintendo Wii*, se adapta perfectamente a las necesidades de cualquier integrante de la familia, desde el archiconocido *Wii Sports*® con capacidad para jugar 8 personas a la vez, pasando por el simulador de fútbol Fifa® ó Pro Evolution Soccer®, el simulador de conducción Need For Speed®, los Simpsons®, los grandes clásicos como el Mario Bross® ó Mario Kart®, los estratégicos como Medal of Honor® ó Splinter Cell® hasta infantiles como pueden ser Pokémon®, Rayman®…etc.

En concreto *Wii Sports* y sus posteriores actualizaciones es el videojuego que lanzó a la fama esta nueva modalidad de entretenimiento en esta sociedad cada vez más preocupada por la forma física y el buen aspecto además respaldado por la gran cantidad de noticias y estudios que alegaban un aumento de enfermedades cardiovasculares y obesidad infantil provocadas por el uso excesivo de estos aparatos.

En definitiva se trata de un dispositivo capaz de entretener de una forma más saludable tanto a mayores como a pequeños ó todos ellos a la vez a un precio asequible.

Figura 2.5: Personas de la tercera edad jugando con *Wii Sports*

Quién iba a decir que una videoconsola se utilice nada más y nada menos que como terapia en diferentes tratamientos, a pesar de las críticas que se han vertido durante años contra ellas, alegando pérdidas de visión, aislamiento, causa de obesidad infantil...

2.3.1. LA WII TERAPIA: RESULTADOS PREVIOS

El soporte de la videoconsola *Wii* como tal se empezó a utilizar hace ya más de dos años en algunas universidades de todo el planeta mediante el desarrollo de proyectos destinados a mejorar las capacidades cognitivas de personas de avanzada edad y mayoritariamente enfocados a enfermos de Alzheimer.

Gracias a estos proyectos e investigaciones se ha demostrado que simplemente mediante el uso de los juegos básicos del software propietario de la *Wii* (golf, tenis, ping-pong, billar, scrabble...) se consiguió mantener e incluso mejorar el estado del paciente en cuanto a procesado de información y coordinación de movimientos se refiere.

En Estados Unidos, concretamente en Florida, en el Communication Disorders Clinic de la University of Central Florida se desarrolló uno de estos proyectos llamado "WII HAB" [3], en el que se puso en práctica el planteamiento anterior consiguiendo unos resultados sorprendentes en los enfermos de Alzheimer que se prestaron a la investigación.

El proyecto fue fundado gracias a subvenciones y donaciones privadas de personas concienciadas con el problema que supone la enfermedad del Alzheimer y posteriormente se mantuvo con un coste de 35$ por persona y día.

Según ha revelado el diario *Times*, el Colegio Médico de Georgia (EEUU) ha anunciado los llamativos resultados de su investigación sobre los beneficios de la Wii en personas con Parkinson [4]. Para esta investigación, dieciocho personas con esta enfermedad jugaron a *Wii Sports* durante una hora al día, tres veces a la semana durante un mes. El resultado: todos los participantes mostraron significantes mejoras en su capacidad motriz, forma física y niveles de energía.

Asimismo, la depresión que suele afectar a más de la mitad de las personas con esta enfermedad, también se aminoró mejorando su estado de ánimo. Boxeo, tenis, bolos… los pacientes se divierten mientras ejercitan su cuerpo. "Puedo decir honestamente que estoy muy sorprendido de las grandes mejoras", declaró al *Times* el Dr. Ben Herz, terapeuta y profesor en la escuela de Ciencias de la Salud del Colegio Médico de Georgia. "La *Wii* permite a los pacientes trabajar en un entorno virtual que es seguro, divertido y motivador".

En el Reino Unido, la psicoterapeuta Rebecca Redmond ha lanzado una comunidad 'on line' para aquellos que utilizan la *Wii* como parte de su programa de rehabilitación (Wii hab) [5]. Redmond trabaja en el National Star College in Cheltenham, donde jóvenes con discapacidades físicas y psíquicas disfrutan jugando como parte de su rehabilitación y afirma que no es un tratamiento como tal, pero que es una más de las herramientas que disponen.

A nivel nacional, es conocido el proyecto que lleva a cabo desde hace años el CRE de Salamanca, en colaboración con el Ministerio de Sanidad y Política Social [6]. Se persigue diseñar, evaluar e implementar una nueva forma de terapia que produzca una mejora y un mantenimiento funcional en las áreas cognitiva y social del enfermo, potenciando las capacidades existentes de la persona con demencia, utilizando para ello la videoconsola *Wii* y el juego Big Brain Academy *Wii Degree*®.

El objetivo principal es probar la eficacia terapéutica de Wii terapia en personas con enfermedad de Alzheimer u otras demencias y de esta forma reducir la carga farmacológica eliminando pastillas en favor de ejercicios saludables.

Específicamente se busca:

- Promoción, mediante el juego, de un estado de bienestar reduciendo las respuestas de ansiedad y depresión, mejorando la calidad de vida.
- Apoyo social. El formato de grupo promociona la interacción de los participantes, actuando éste como grupo de soporte y autoayuda.
- Estimulación de las funciones cognitivas superiores como la atención, percepción, memoria, orientación, cálculo, razonamiento y lenguaje.

2.3.2. EL WIIMOTE COMO RATÓN PARA DISCAPACITADOS

El 27 de noviembre de 2008 salió a la luz la noticia de que un grupo de investigadores de la Universidad Politécnica de Madrid (UPM) convirtieron el mando de la consola *Wii* en una herramienta para personas discapacitadas [7].

La idea, que aún se encontraba en fase de laboratorio, consiguió el reconocimiento de la comunidad científica y de los interesados, para quienes supone una notable reducción de precio respecto a aparatos de características similares.

La razón de escoger el mando de la *Wii* fue su disponibilidad en el mercado y su bajo precio, frente a otros aparatos dirigidos específicamente a las personas discapacitadas,

ya que, según afirman, existen sistemas que permiten manejar el ratón con la mirada, pero cuestan 6.000 euros, mientras que el *Wiimote*, 40 euros.

Los investigadores se han centrado en personas con Parkinson, movilidad reducida en los brazos, visión borrosa y tetraplejia, haciendo algunas modificaciones al *Wiimote* creando complementos de bajo coste, tales como diademas, pulseras, gafas o cualquier tipo de soporte fruto del ingenio para conseguir adaptarse a casi cualquier tipo de discapacidad.

Para personas con problemas de visión y deficiencias cognitivas, el *Wiimote* convierte la pantalla normal o proyectada en la pared en una pantalla táctil, más fácil de ver y manejar.

En cuanto a las personas con movilidad reducida en los brazos, Parkinson o en silla de ruedas, pueden usar la parte del cuerpo donde tengan más control para accionar uno de los inventos del equipo.

2.3.3. GORRA CON WIIMOTE INCORPORADO PARA DISCAPACITADOS

En la Universidad Carlos III de Madrid se ha desarrollado un interfaz que permite a los pacientes usar los movimientos del cuello para controlar aplicaciones en el ordenador. Se basa en el mando Wiimote acoplado a una gorra y es de gran utilidad para personas con algún grado de discapacidad en manos y brazos [8].

Figura 2.6: Gorra con *Wiimote* acoplado en la visera

Por otro lado, la Universidad de Lleida ha desarrollado una herramienta denominada "Head-Mouse" que permite controlar el puntero del ordenador con movimientos de cabeza [9].

La aplicación es gratuita y está disponible 'on line', pudiéndose usar simplemente disponiendo de una cámara web. También se recomiendan cámaras y se explica la configuración óptima para Windows.

Definitivamente, esta línea de investigación que se ha seguido para conseguir dar más libertad a personas con discapacidad ha sido la inspiración para el desarrollo del módulo que se pretende implementar en este proyecto.

2.4. TABLA WII BALANCE BOARD

Este periférico de la videoconsola es una tabla inalámbrica capaz de detectar el peso del jugador así como la presión en su superficie mediante los sensores incorporados en el interior.

Figura 2.7: Tabla *Wii Balance Board*

Junto al software necesario, el llamado *Wii Fit*® nos propone tonificar nuestro cuerpo mediante ejercicios divididos en varias áreas (tonificación, yoga, aeróbic y equilibrio), que dependiendo del nivel de dificultad pueden resultar muy complejos.

Figura 2.8: Personas con movilidad reducida usando la *Wii Balance Board*

Es cierto que *Wii Fit* no es un sustituto natural de un gimnasio o un entrenador personal y que no obra milagros, pero nos insta a crearnos un objetivo en materia de reducción de peso para alcanzar un índice de masa corporal cada vez más óptimo, que acompañado de una buena alimentación y la lucha contra el sedentarismo dará siempre resultados positivos.

Figura 2.9: Ejemplos de ejercicios de tonificación muscular

Aparte de los ejercicios propuestos por *Wii Fit*, existen otros juegos que requieren el uso de la tabla de forma que el usuario se encuentra sentado sobre ella (véase disposición en la figura 2.10.

En cuanto a la parte técnica, la *Wii Balance Board* es un dispositivo que alberga cuatro sensores bastante avanzados a un precio reducido de apenas 90 € y que son capaces de sincronizarse con los movimientos del mando *Wiimote* del cual se explicarán sus características en la parte de hardware utilizado para el proyecto, e incluso con el periférico "nunchuck" o "nunchaku".

Estos cuatro sensores se disponen cada uno en una esquina de la tabla de forma que pueden medir el peso y el balanceo de nuestro centro de gravedad al deformarse la parte metálica de superficie en mayor o menor grado. Están fabricados en una aleación de aluminio usada en aeronáutica y son capaces de medir variaciones desde 500 gr hasta 100 kg con una precisión asombrosa teniendo en cuenta de la deformación máxima es

de apenas 0,1 mm según afirma el vocal de la empresa que lo fabrica, Akira Sato at Minebea Co. Ltd.

Figura 2.10: Uso de tabla *Wii Balance Board* sentado

A modo de curiosidad, se pueden encontrar en internet modificaciones tan espectaculares como la realizada sobre unas zapatillas Nike con los cuatro sensores de la tabla incorporados en las plantillas para jugar al *Wii Fit,* véase la figura 2.11. Enlace:

http://hacknmod.com/hack/sneakers-wii-balance-board-nike-wiis/

Figura 2.11: Nike modificadas para simular *Wii Balance Board*

2.5. INTERACCIÓN GESTUAL: NINTENDO DS2 y 3DS

La videoconsola portátil *DS2* de *Nintendo* es un dispositivo que combina la jugabilidad de una consola convencional con la interacción de nuestros movimientos para ofrecer una nueva dimesión de juego.

Aprovechando la tecnología que permite la detección del movimiento se han desarrollado varios programas y aplicaciones de software libre para hacer compatible este dispositivo con un ordenador con el sistema operativo Linux instalado.

Aplicado al campo que se trata en este proyecto, es interesante mencionar la aplicación desarrollada para sincronizar el escritorio del ordenador y visualizarlo en la pantalla superior de la *DS2*, del cual existe un vídeo demostrativo que se puede visualizar en el enlace:

http://www.youtube.com/watch?v=SyZ82zmA5xo

A partir de esta aplicación se pueden desarrollar muchas más, como el uso de programas específicos de Linux adaptados a la videoconsola e incluso sería interesante la generación de un software capaz de monitorizar el este escritorio remoto la información de que envían los sensores de movimiento para su posterior procesado.

Se puede verificar todo el proceso para configurar el escritorio remoto en el siguiente enlace:

http://crysol.org/es/node/622

Por otro lado, la nueva *Nintendo 3DS* (figura 2.12), que es a su vez la evolución de la *DS2*, se ha confirmado como la consola portátil multimedia en la que se podrán ver películas en 3D, aparte una buena colección de videojuegos y multitud de opciones online.

Su pantalla superior es de 3.53 pulgadas con tecnología LCD de 800x240 píxeles y no es necesario usar gafas de 3D para disfrutar de la experiencia y la inferior es de 3.02 pulgadas, táctil y con 320x240 píxeles de resolución aparte de una cámara interna y dos externas.

Las cámaras exteriores son capaces de tomar imágenes que se encuentren en frente del dispositivo y mostrarlas en la pantalla en 3D mientras que la interior sirve para capturar al usuario e interactuar con él al estilo de lo que ya hacía la anterior *DSi*.

Figura 2.12: La nueva *Nintendo 3DS*

CAPÍTULO 3. ACONDICIONAMIENTO DEL MÓDULO PROPIO

Según se ha detallado anteriormente, el módulo que se ha diseñado está englobado en el proyecto ASIBOT desarrollado por el grupo de investigación Robotics Lab del Departamento de Ingeniería de Sistemas y Automática de la UC3M [10].

3.1. BASE EXISTENTE

Para conseguir crear un entorno de trabajo cómodo y accesible por parte de todos los alumnos, investigadores y desarrolladores, existen una relación de estándares utilizados que es importante respetar (RCGv03) [11].

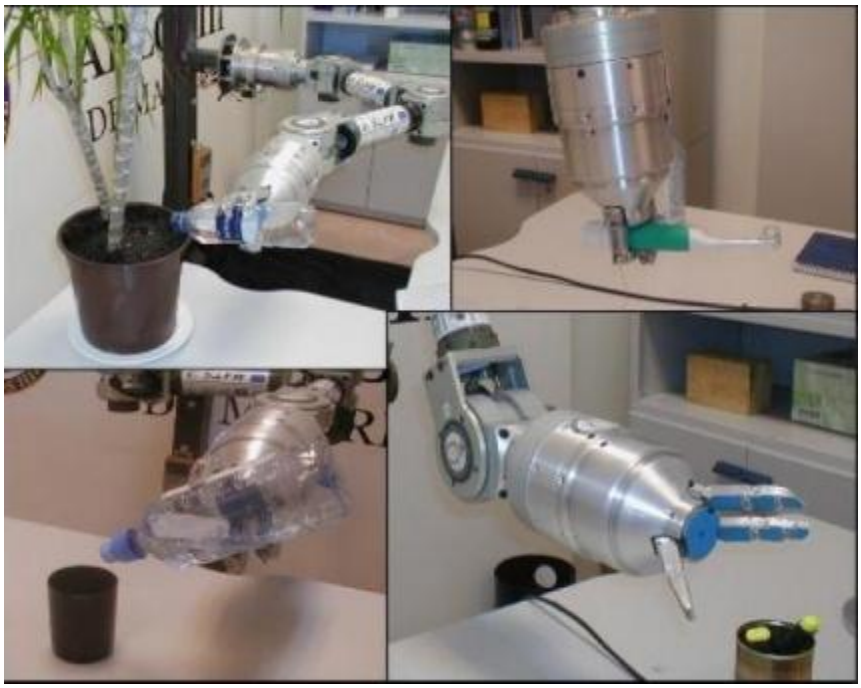


Figura 3.1: Robot asistencial de cocina ASIBOT. Imagen Wiki Robotics Lab

Toda esta información se encuentra perfectamente documentada en la Wiki de robots de la UC3M, dentro del apartado ASIBOT, accesible a través del enlace:

http://robots.uc3m.es/w/index.php/Main_Page

Siguiendo las recomendaciones que aquí se encuentran, se procedió a instalar los paquetes necesarios para la adaptación del módulo propio.

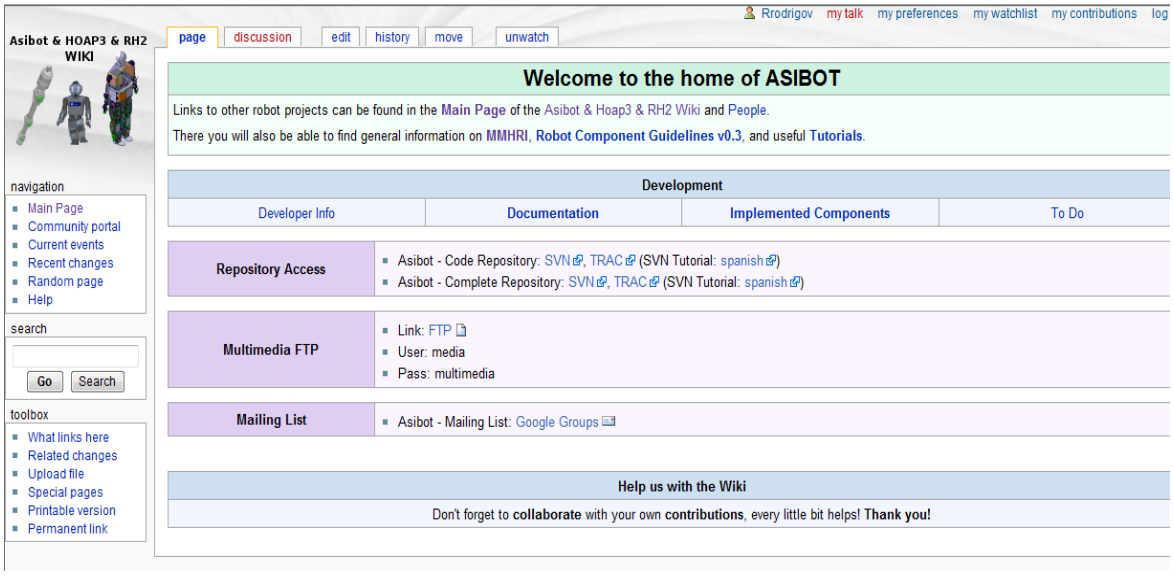


Figura 3.2: Wiki del ASIBOT

En las siguientes líneas se pretende resumir el contenido de dicha Wiki para su rápida comprensión.

3.1.1. ROBOT COMPONENT GUIDELINES V0.3

RCGv03 tiene 4 aspectos fundamentales que son: formato de comandos, estructura de carpetas, versión de librerías y nombramiento de módulos.

Estas directivas son necesarias para compatibilizar los distintos módulos que se han desarrollado y que se desarrollarán en un futuro, siendo necesario incluir en el repositorio las novedades que se utilicen por primera vez en un módulo.

Dicho repositorio de Robotics Lab perteneciente al desarrollo del ASIBOT está basado en Subversion, que es una herramienta software para la gestión de ficheros online y se puede encontrar un tutorial de uso en español en la dirección:

http://asrob.uc3m.es/w/index.php/Tutorial_SVN

En primer lugar es importante comprobar las dependencias que se van a tener respecto a las librerías y comprobar si se tiene instalada en el ordenador la versión indicada.

A continuación es recomendable echar un vistazo a los nombres utilizados para definir las variables y si es posible utilizar las disponibles en el repositorio.

El siguiente paso es crear el proyecto en cuestión en la rama del repositorio en la cual estemos trabajando y se puede empezar a generar código, pero respetando el formato de comando que corresponda a la categoría del módulo propio.

En los siguientes puntos se detallan las directivas de las que se ha hecho referencia anteriormente y las utilizadas en este módulo.

3.1.1.1. COMMAND FORMAT

El formato de comandos del módulo propio es el correspondiente a la de la categoría:

Module: hmi_name

Tal denominación se debe a que tras discutir la naturaleza del proyecto se decidió que se podía considerar más que un sensor, como un interfaz hombre-máquina, ya que a través del mando de la *Wii*, que contiene los sensores, somos capaces de visualizar nuestros movimientos en la pantalla de un ordenador, por lo que se englobó en la categoría de Human Machine Interfaces (HMI).

3.1.1.2. FOLDER STRUCTURE

Se recomienda la siguiente estructura de carpetas de proyecto:

```
| -- AUTHORS
| -- CmakeFiles.txt
| -- config
| -- doc
|    `-- Doxyfile
|-- INSTALL
|-- out
|     `-- share
`- src
    | -- .cpp
    `-- .h
```

Las cuales se irán completando a medida que se desarrolla el módulo.

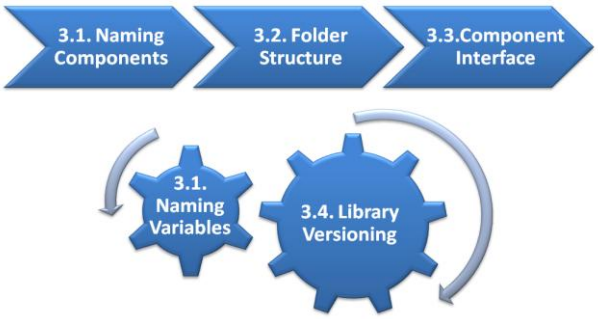


Figura 3.3: RCGv03. Metodología para creación de componentes

3.1.1.3. LIBRARY VERSIONING

En el repositorio se encuentran la mayoría de las librerías necesarias para el correcto funcionamiento de los programas a desarrollar, siendo recomendable incluir cualquier librería usada por primera vez en dicho repositorio.

YARP 2.2.6

Esta es la librería fundamental para que sea posible la utilización del módulo en cualquier ordenador. Seguimos los pasos indicados para instalarla en el equipo:

```
wget http://downloads.sourceforge.net/yarp0/yarp-2.2.6.tar.gz
tar -zxvf yarp-2.2.6.tar.gz
cd yarp-2.2.6; mkdir build; cd build
cmake ..
make
sudo make install
cd ../..
```

A su vez, YARP depende de ACE 5.7.1, y de la misma forma en un terminal se teclea:

```
wget http://download.dre.vanderbilt.edu/previous_versions/ACE-5.7.1.tar.gz
tar -zxvf ACE-5.7.1.tar.gz
cd ACE_wrappers; mkdir build; cd build
../configure --disable-ssl --disable-ace-examples
make -j3
sudo make install
cd ../..
```

En cuanto a la parte visual, para no tener problemas a la hora de compilar con CMake, es necesario instalar libgtk2.0-dev, que es la versión más actualizada.

En un terminal: sudo apt-get install libgtk2.0-dev

El resto de las librerías que se han utilizado se detallarán más adelante debido a que no son estrictamente necesarias para la compatibilidad de los módulos, sino que se descargan directamente con el programa utilizado.

3.1.1.4. NAMING MODULES

Las directrices para nombrar el módulo y las variables dentro del HMI (Human Machine Interfaces) que se está desarrollando para asegurar la compatibilidad son:

- Cada módulo tendrá un prefijo de tres letras que indicará la naturaleza del mismo. En este caso "hmi_". A continuación una descripción del mismo, concretamente "hmi_wiimote".
- Cada puerto contendrá una descripción de la implementación del módulo y una pequeña nomenclatura que lo identifique. Los usados en este módulo son (en hmi_wiimote.c):

```
Línea 206    BufferedPort<Bottle>miPuerto;
Línea 1165   miOutput.clear ( );
Línea 1178   miPuerto.write ( );
```

- Cada variable interna relacionada con algún aspecto comunicativo contendrá: un descriptor (bottle, int..), un pequeño nombre que identifique el tipo de dato y un identificador de su naturaleza I/O (siendo I para input y O para output, ó IO para input-output). En este caso:

```
Línea 214    miPuerto.open("/wiimote_info");
Línea 1165   Bottle& miOutput = miPuerto.prepare ( )
             miOutput.addString ("accX");
             miOutput.addDouble (a_x);
             miOutput.addString ("accY");
             miOutput.addDouble (a_y);
             miOutput.addString ("accZ");
             miOutput.addDouble (a_z);
             miOutput.addString ("roll");
             miOutput.addDouble (roll);
             miOutput.addString ("pitch");
             miOutput.addDouble (pitch);
```

La inclusión de las líneas anteriores se detalla en el apartado de puesta en marcha, ya que son el mecanismo de intercambio de información de la librería YARP.

3.2. SOFTWARE NECESARIO

Una vez que llega la hora de generar código, son necesarias diversas aplicaciones, drivers, librerías…. para que sea posible crear el módulo deseado.

El los puntos siguientes se detallan todas las herramientas utilizadas de software, de lo general a lo particular, es decir, empezando por el sistema operativo UBUNTU, pasando por los programas necesarios para el desarrollo del proyecto así como las librerías que permiten su funcionamiento.

3.2.1. UBUNTU 10.04 LTS

Antes de empezar a desarrollar el proyecto en cuestión y tras seguir varias recomendaciones para hacerlo más llevadero, se decidió usar este sistema operativo por su versatilidad y comodidad a la hora de programar.

Ubuntu es un sistema operativo de código abierto desarrollado en torno al kernel Linux, cuya filosofía se basa en los siguientes principios: software gratuito, disponible en la lengua materna del usuario y personalizable a cualquier nivel según las propias preferencias.

Según las anteriores premisas parece el software perfecto para empezar a trabajar, por lo tanto, accediendo a la página www.ubuntu.com se puede descargar la última actualización del software de manera totalmente gratuita e instalarla en el ordenador.

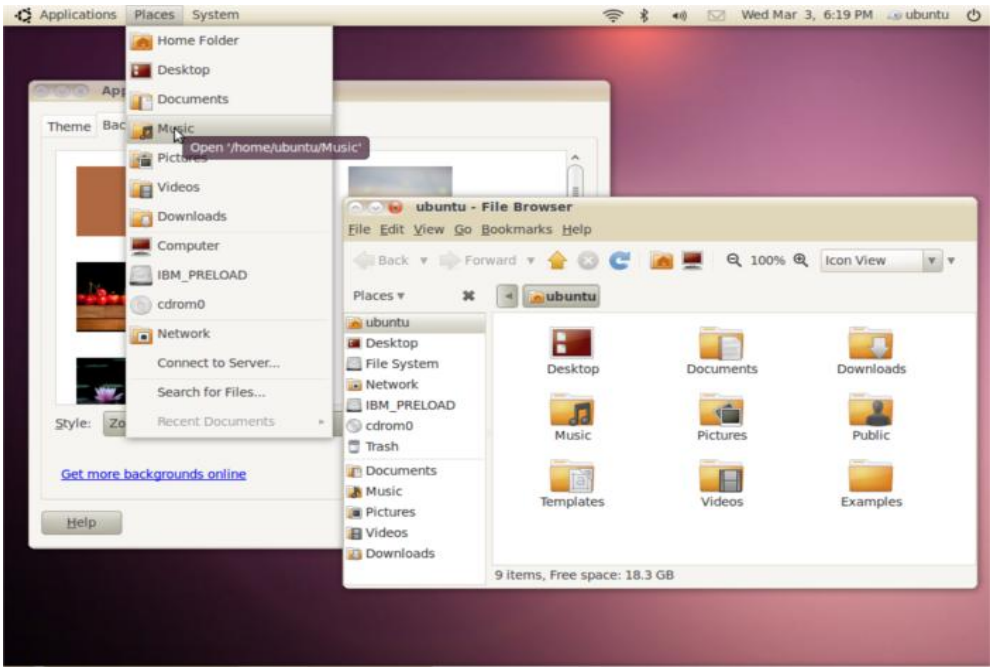


Figura 3.4: Escritorio de trabajo de Ubuntu con varias aplicaciones abiertas

3.2.2. CMAKE 2.6

CMake "cross platform make", es un sistema multiplataforma para realizar compilación automática es decir, una herramienta para generar makefiles donde el usuario únicamente debe crear un archivo de configuración con todo lo que necesite: nombre del ejecutable, ficheros cabecera, ficheros fuente, enlazar librerías, etc.

En el caso de CMake, este fichero se llama CMakeLists.txt.

También, si se tiene un proyecto dividido en subdirectorios, por cada módulo/librería generada por nosotros podemos tener en cada subdirectorio un fichero CMakeLists.txt para obtener una mejor organización de las necesidades de nuestro proyecto.

Para instalarlo en el ordenador, se teclea en un terminal:

```
sudo apt-get install cmake
```

Con el siguiente comando podemos saber qué versión de CMake se ha instalado, en nuestro caso la 2.6, sin embargo en noviembre de 2009 se publicó la 2.8 que es la más actualizada hasta el momento.

```
cmake --version
```

A continuación, en el directorio que se ha originado, creamos el fichero de configuración de CMake (CMakeLists.txt) y que contiene lo siguiente:

Declaración del nombre del proyecto:

```
PROJECT(hmi_wiimote.cpp)
```

Se crea un conjunto donde se encuentran todos los códigos fuente:

```
SET(SRCS hmi_wiimote.cpp)
```

Definición del ejecutable:

```
ADD_EXECUTABLE(hmi_wiimote ${SRCS})
```

Se buscan los paquetes necesarios para compilar (bibliotecas):

```
FIND_PACKAGE(wxWidgets COMPONENTS core base REQUIRED)
```

Que luego se enlaza con el proyecto de la siguiente forma:

```
TARGET_LINK_LIBRARIES(hmi_wiimote ${wxWidgets_LIBRARIES})
```

Después se definen en CMAKE_CXX_FLAGS los flags que se necesiten para compilar:

```
SET_TARGET_PROPERTIES(hmi_wiimote PROPERTIES COMPILE_FLAGS “-g -Wall `wx-config –libs` `wx-config –cxxflags`”)
```

Una vez configurado el fichero CMakeLists.txt se ejecuta en el terminal el siguiente comando para generar el makefile:

```
cmake CMakeLists.txt
```

Con estos pasos se generarán automáticamente muchos ficheros .cmake que no se modificarán de momento.

3.2.3. WIIUSE V0.12 en C

Wiiuse es una librería escrita en C capaz de conectarse con los controladores de varios aparatos de la videoconsola *Nintendo Wii* como el *Wiimote*, el mando de control clásico, la guitarra de *Guitar Hero 3*® etc, a través de un dispositivo de *Bluetooth* convencional compatible.

Se puede descargar de forma totalmente gratuita de la misma página del creador e instalarla en el ordenador. Enlace:

http://wiiuse.net/

En este proyecto es necesaria para el funcionamiento del Wiimote_Server_Module el cual se analiza en la subsección 3.2.4.

3.2.4. WIIMOTE SERVER MODULE

Wiimote_server_module es un módulo creado por Eric Sauser que, como su nombre indica, se conecta al *Wiimote* mediante el uso de la librería Wiiuse previamente descargada en el ordenador y esta creado para enviar los datos mediante YARP, por lo que pareció muy útil a primera vista.

Se puede descargar gratuitamente de la página del creador:

http://eris.liralab.it/iCub/main/dox/html/group__icub___lasaImitation__WiimoteServerModule.html

Además tiene dos modos de funcionamiento, que se pueden seleccionar y modificar fácilmente por código:

- Event Mode: envía un dato cada vez que detecta un cambio en alguna variable (write strict). Cuando éste sea “false”, entonces se estará enviando información en modo streaming.
- Streaming Mode: envía continuamente información sobre el estado de las variables (write).

Debido a las necesidades del módulo hmi_wiimote el modo streaming es el más útil porque se necesita continuamente la información del estado las variables del acelerómetro y el sensor giroscópico (X, Y, Z, roll y pitch) por lo que al ejecutar el Wiimote_Server_Module hay que seleccionar el modo con una “S”.

3.2.5. CWIID 0.6.00

CWiid es una colección de herramientas que sirven de interfaz entre diversos controladores de la videoconsola *Nintendo Wii* y un ordenador convencional a través de un dispositivo *Bluetooth* convencional compatible.

Básicamente es lo mismo que la librería Wiiuse pero CWiid es más completo ya que incluye una visualización muy clara de las variables y los movimientos, en este caso del *Wiimote*, que es de gran ayuda.

CWiid incluye en su paquete la librería "libcwiid", la API necesaria para las comunicaciones y el programa de visualización Wmgui.

Para instalar el paquete completo, directamente desde la pagina web http://abstrakraft.org/cwiid/, se descarga y se instala en el ordenador siguiendo el procedimiento habitual.

En "libcwiid" se encuentra la relación de funciones y comandos gracias a los cuales es posible su funcionamiento. Si es necesario pueden ser modificados para cambiar el funcionamiento pero en este caso no se alterarán, ya que la parte que interesa modificar es la de obtención de datos, y no la de comunicaciones, la cual se realizará mediante YARP como se documentará más adelante.

3.2.6. WMGUI

Como se ha mencionado anteriormente, Wmgui es una interfaz (visualización) de los botones y las aceleraciones de los distintos periféricos de la videoconsola *Wii*.

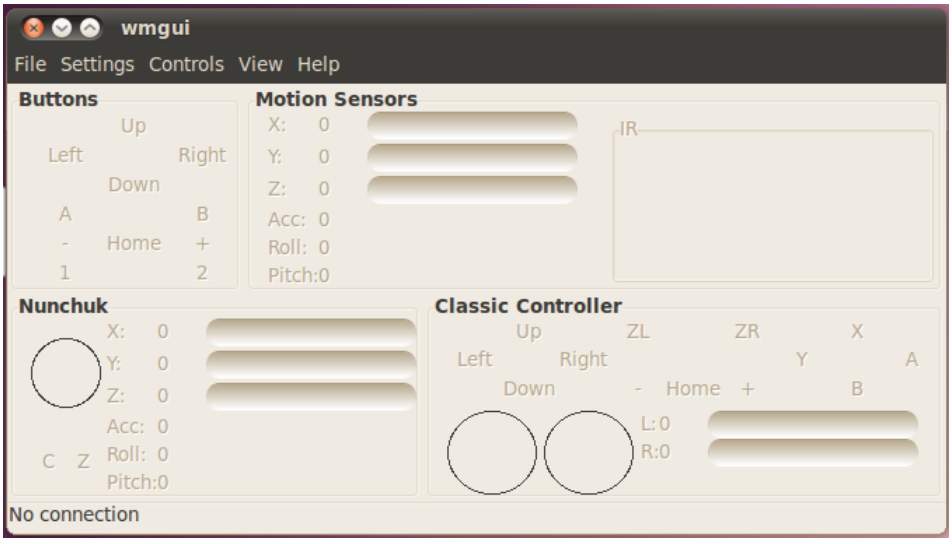


Figura 3.5: Interfaz Wmgui previa a la conexión con el mando

Una vez instalada, se conecta el dispositivo *Bluetooth* al ordenador mediante el puerto USB y en el menú "file" se selecciona "connect".

Siguendo la indicación del programa, se presionan los botones 1 y 2 en el mando y se pulsa en "aceptar".

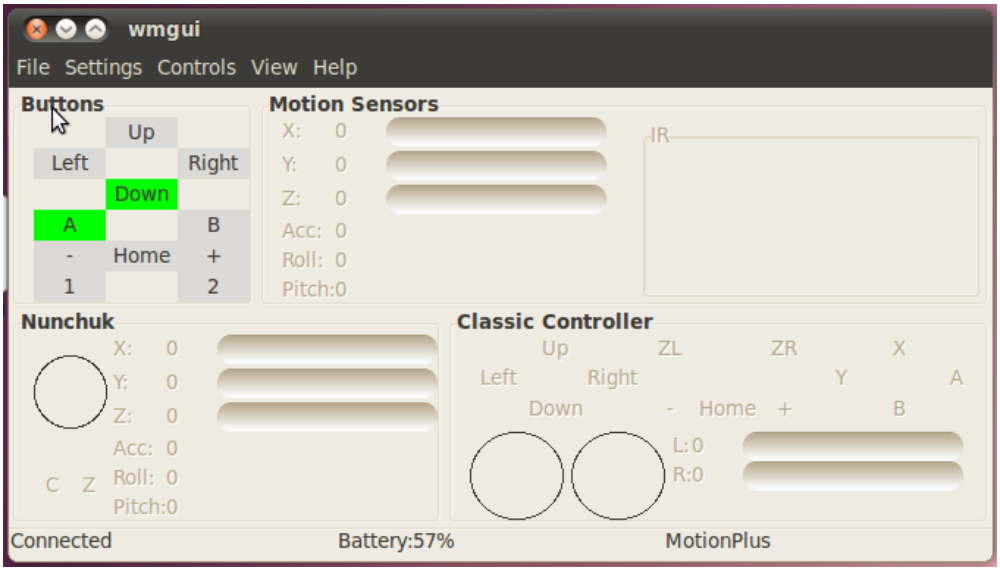


Figura 3.6: Habilitada la detección de los botones

Gracias a este interfaz ahora se puede visualizar en el ordenador si está pulsado algún botón del mando, pero si además se necesita información de las aceleraciones que actúan sobre él al moverlo con la mano, en el menú "settings" se escoge la opción "acc Data" que activa las 3 barras correspondientes a los ejes en el espacio (x, y, z) y muestra la aceleración que se está ejerciendo en cada momento en cada uno de dichos ejes.

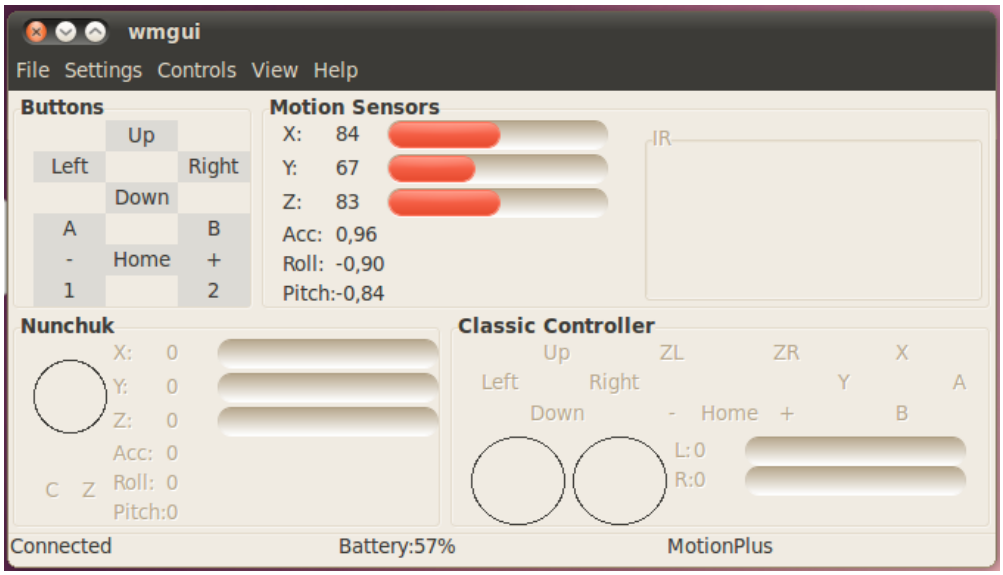


Figura 3.7: Habilitada la detección del acelerómentro en los ejes X, Y y Z.

Si se ha llegado hasta este punto, quiere decir que CWiid funciona correctamente en nuestro ordenador y que el dispositivo *Bluetooth* escogido es compatible con el *Wiimote*.

Más adelante, en el apartado de *Bluetooth* se exponen los requisitos que debe cumplir este dispositivo para su conexión.

3.3. HARDWARE UTILIZADO

Como se ha mencionado anteriormente, para el funcionamiento de este módulo son necesarios tres elementos: un ordenador con Linux instalado, un mando *Wiimote Plus* de *Nintendo* y un dispositivo USB de *Bluetooth* compatible.

En los siguientes apartados se detalla el hardware utilizado, sus especificaciones y el proceso necesario para su correcto funcionamiento.

3.3.1. MANDO NINTENDO® WIIMOTE PLUS

El accesorio *Wii Motion Plus* anunciado el 14 de julio de 2008 y presentado en Madrid por *Nintendo* el 16 de julio de 2009 volvió a revolucionar el sistema de control para videojuegos al reflejar de forma más rápida y precisa los movimientos en un espacio tridimensional.

Wii Motion Plus se conecta en la parte inferior del mando original de *Wii* y, combinado con el acelerómetro y la barra de sensores del original, registra la posición y orientación del brazo del jugador de una manera más detallada, ofreciendo así un nivel de precisión e inmersión nunca visto.

Sus dimensiones son 14,6x3,5x3,1 cm a las que hay que sumar las del accesorio *Wii Motion Plus*, de forma que las dimensiones totales son 19,7x3,5x3,1 cm y un peso aproximado de 300gr.

Cada uno de los movimientos que el jugador haga con el brazo o con la muñeca, por ligero que sea, se reproducirá al detalle en pantalla en tiempo real, creando así una respuesta de 1:1 en la partida.

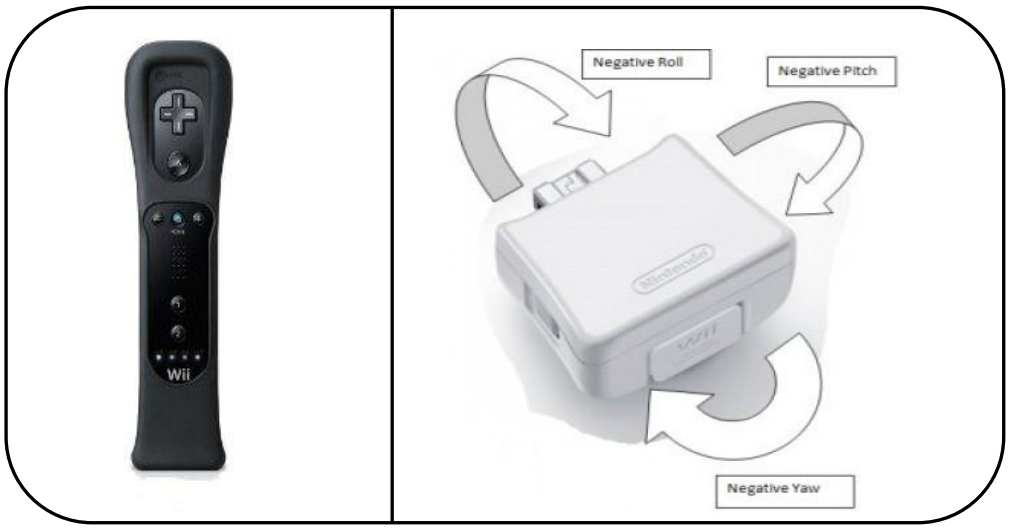


Figura 3.8: Mando *Wii Remote* con accesorio *Wii Motion Plus*

Técnicamente se trata de un sensor giroscópico (MEMS MPU 3000) que detecta la rotación y la elevación (roll y pitch) y permite mediante el software apropiado dar una posición exacta en tres dimensiones, pero depende, eso sí, del acelerómetro (modelo ADXL330) del mando original que proporciona las aceleración en las coordenadas X, Y, Z.

Juntos son capaces de detectar hasta 1600 ángulos de giro por segundo, lo que supone una precisión asombrosa para un dispositivo de reducido coste.

No se profundizará en detalles debido a su complejidad de funcionamiento y desarrollo, pero se encuentra documentado en la página:

http://en.wikipedia.org/wiki/Vibrating_structure_gyroscope#Tuning_fork_gyroscope

Figura 3.9: Uso del mando *Wiimote Plus*

Para detectar el punto al que se está dirigiendo el mando en cualquier instante se usa un sensor óptico PixArt®, pero en vez de detectar los colores de la pantalla como antes, ahora se detecta la barra de LEDs infrarrojos que se coloca en la parte superior de ésta (modelo RVL-014).

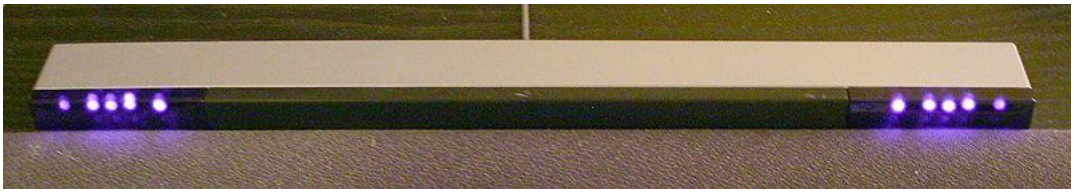


Figura 3.10: Barra de LEDs y sensor necesario para la recepción de datos

En este proyecto, en vez de usar el software original de *Nintendo*, se ha utilizado una adaptación para Linux llamada CWiid, la cual funciona al igual mediante *Bluetooth* para enviar los datos al ordenador y se visualizan en un pequeño interfaz llamado Wmgui (3.2.6.), pero no se utilizan el sensor óptico ni la barra de LEDs por la sencilla razón de que no son relevantes para el funcionamiento del módulo *Wiimote* que se está desarrollando.

En cuanto a la memoria, El *Wiimote* contiene un chip de 16 kB EEPROM donde una sección de 6 kilobytes puede ser libremente leída y escrita por el host.

Parte de esta memoria está disponible para almacenar hasta 10 avatares Mii (caracterización de un personaje para cada jugador), que pueden ser transportados para su uso en otra videoconsola *Wii*. Al menos 4000 bytes están disponibles y no utilizados antes de los datos Mii.

La alimentación se proporciona mediante dos baterias AA como fuente de energía, que pueden alimentar el *Wii* remote durante 60 horas usando sólo la función de acelerómetro y 25 horas utilizando acelerómetro y puntero.

Todavía no se ha lanzado un cargador para el *Wiimote*, pero terceros fabricantes del mercado han desarrollado sus soluciones. Además, un condensador 3300 μF proporciona una fuente temporal de energía con movimientos rápidos del *Wiimote*, cuando la conexión a las pilas pudiera ser interrumpida temporalmente.

Durante el período de prueba del mando para el desarrollo del proyecto apenas gastó un 55% de la carga de las pilas, según muestra el interfaz Wmgui en la parte inferior, por lo que se puede deducir que su duración es más que aceptable.

Figura 3.11: Circuito interno del *Wiimote*

Figura 3.12: Accesorio Wii Motion Plus

3.3.2. DISPOSITIVO BLUETOOTH USB

En primer lugar, es importante saber qué es *Bluetooth* y cómo funciona.

Se denomina *Bluetooth* al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, con una cobertura baja y basada en transceptores de bajo coste.

Gracias a este protocolo, los dispositivos que lo implementan pueden comunicarse entre ellos cuando se encuentran dentro de su alcance.

Las comunicaciones se realizan por radiofrecuencia en la banda ISM a 2.4 GHz de frecuencia de forma que los dispositivos no tienen que estar alineados y pueden incluso estar en habitaciones separadas si la potencia de transmisión lo permite. Estos dispositivos se clasifican como "Clase 1", "Clase 2" o "Clase 3" en referencia a su potencia de transmisión, siendo totalmente compatibles los dispositivos de una clase con los de las otras. En la tabla 3.1 se muestra dicha clasificación.

Tabla 3.1: Potencia máxima permitida en dispositivos *Bluetooth*

Clase	Potencia máxima permitida	Potencia máxima permitida	Rango (aprox)
1	100 mW	20 dBm	100 m
2	2.5 mW	4 dBm	25 m
3	1 mW	0 dBm	1 m

El hardware que compone el dispositivo *Bluetooth* está compuesto por dos partes:

- Un dispositivo de radio, encargado de modular y transmitir la señal

- Un controlador digital, compuesto por una CPU, por un procesador de señales digitales (DSP - Digital Signal Processor) llamado Link Controller (o controlador de Enlace).

El LC o Link Controller está encargado de hacer el procesamiento de la banda base y del manejo de los protocolos ARQ y FEC de capa física. Además, se encarga de las funciones de transferencia (tanto asíncrona como síncrona), codificación de Audio y cifrado de datos.

El CPU del dispositivo se encarga de atender las instrucciones relacionadas con *Bluetooth* del dispositivo anfitrión, para así simplificar su operación. Para ello, el CPU utiliza un software denominado Link Manager que tiene la función de comunicarse con otros dispositivos por medio del protocolo LMP.

Para la comunicación con el *Wiimote* se ha utilizado un dispositivo 1.2 según se especifica en la tabla 3.2, aunque el más utilizado actualmente es el 2.0, pero no el más potente.

Tabla 3.2: Versiones de dispositivos *Bluetooth* en el mercado y ancho de banda

Versión	Ancho de banda
Versión 1.2	1 Mbit/seg
Versión 2.0	3 Mbit/seg
Versión 3.0	24 Mbit/seg

Esto se debe fundamentalmente por el temor a compatibilidades con Linux.

Es un dispositivo un tanto obsoleto pero que funcionó sin problemas al probarlo por primera vez conectando el mando *Wiimote Plus* con el ordenador y ejecutando el programa Wmgui que se descargó junto con la librería CWiid.

El dispositivo seleccionado es del fabricante *ZOOM*®, concretamente el modelo 4310 de la “series 46”.

3.4. FUNCIONAMIENTO DEL MÓDULO HMI_WIIMOTE

La figura 3.13 es un diagrama explicativo del funcionamiento interno del módulo Hmi_wiimote y de su conexión con el módulo creado por Daniel García Sánchez.

Se ha creado un pequeño programa de ejecución por lotes llamado "conecta" que sirve para simplificar las conexiones de ambos módulos y efectuarlas automáticamente en vez de indicarlas cada vez que se quiera ejecutar el programa en el terminal. El contenido de dicho programa se encuentra documentado en la subsección 5.3.1.

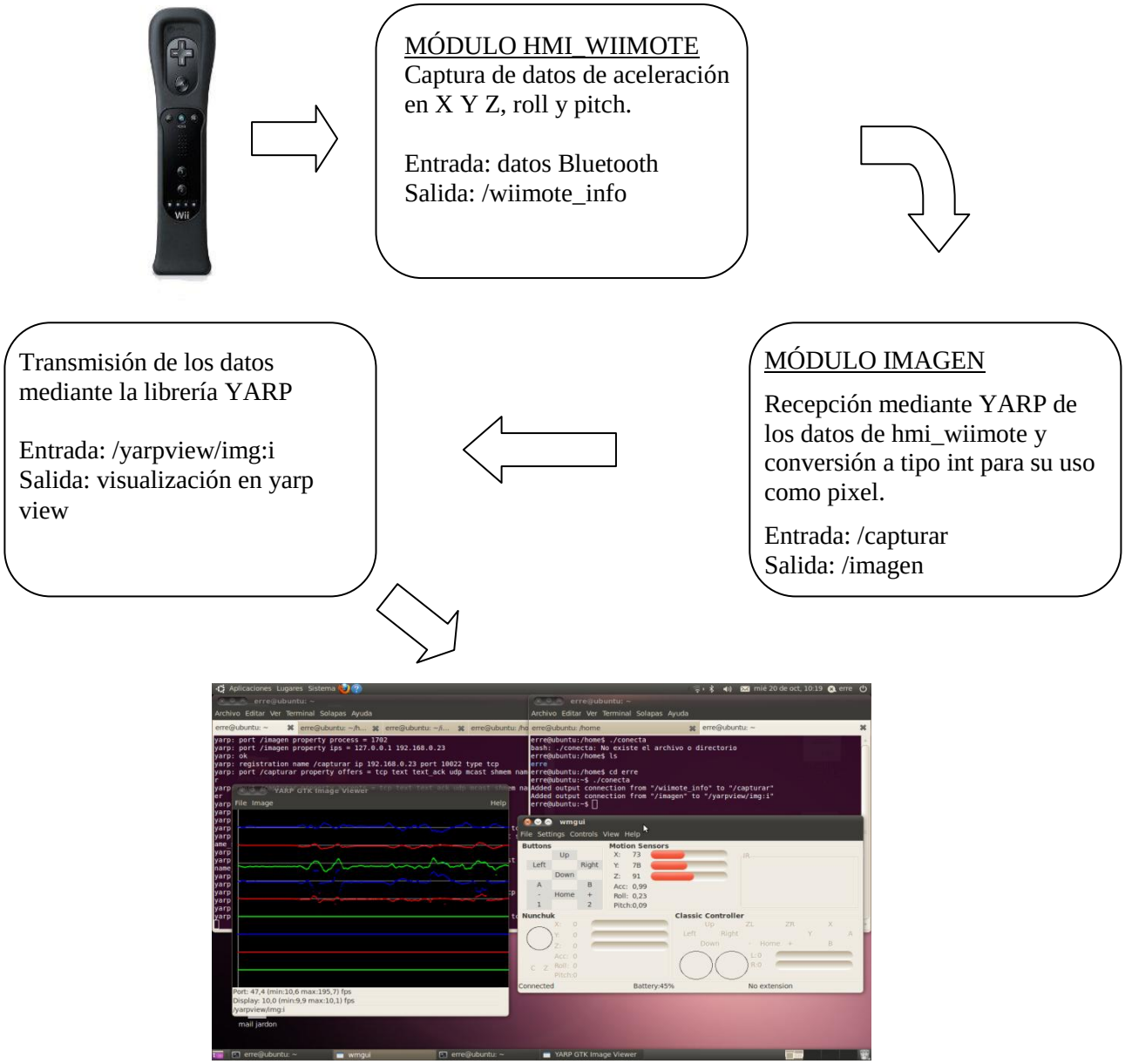


Figura 3.13: Diagrama de bloques del funcionamiento del módulo Hmi_wiimote

En primer lugar se capturan los datos procedentes del mando *Wiimote Plus* mediante el programa Wmgui y la correspondiente librería CWiid que le permite funcionar correctamente. Se consigue gracias a que el periférico de *Nintendo* es compatible con el protocolo *Bluetooth* y basta con conectarlo al puerto USB del ordenador para que Wmgui capture los datos de las aceleraciones en X, Y y Z, además de "roll" y "pitch" según se muestra en el primer bloque.

Una vez capturadas las variables, son mostradas en la pantalla de un terminal, y según se explicará mas adelante, son enviadas por un puerto YARP que se abre automáticamente al ejecutar el módulo Hmi_wiimote con el que se está trabajando.

En segundo lugar, los datos que se han enviado mediante YARP, son capturados por el módulo "imagen", en el cual son tratados de forma que se lean correctamente en el siguiente bloque coincidiendo con sus especificaciones. De nuevo se abre un segundo puerto YARP y se envían los datos procesados por el módulo "imagen" a un visualizador de esta misma librería.

El resultado de esta operación es la pantalla con fondo negro y líneas en azul, rojo y verde que representan la variación de las cinco variables inicialmente capturadas por el módulo Hmi_wiimote.

Se puede comprobar el la figura 5.2 de la sección 5.3 la disposición de las variables en la visualizador "yarpview".

CAPÍTULO 4. DESARROLLO DEL MÓDULO PROPIO (HMI_WIIMOTE)

En los siguientes apartados se expone la metodología que se ha seguido a nivel software para confeccionar el módulo Hmi_wiimote del que se ha hablado anteriormente.

Existen varias formas de transmitir la información que proporciona el mando *Wii Motion Plus* a un ordenador por lo que se probó las más completas hasta dar con la adecuada para las necesidades de información.

En este caso, los datos que se necesitan son las variables de aceleración en los tres ejes del espacio X, Y y Z y las proporcionadas por el accesorio *Plus*, "roll" y "pitch".

Tras realizar una búsqueda exhaustiva en internet sobre distintos métodos para obtener los datos, se encontraron dos variantes que aparentemente resultaban efectivas y sencillas de usar:

- Librería Wiiuse y Wiimote_Server_Module para transmisión de datos

  http://www.wiiuse.net/

  http://eris.liralab.it/iCub/main/dox/html/group__icub____lasaImitation__WiimoteServerModule.html

- Librería CWiid e interfaz Wmgui

  http://abstrakraft.org/cwiid/

El primer intento se realizó con Wiiuse pero los resultados obtenidos no eran favorables según se explica en el punto 4.1. y se utilizó finalmente CWiid.

En los siguientes apartados se desarrolla todo el proceso de comprobación y funcionamiento.

4.1. PRUEBA 1. WIIUSE Y WIIMOTE_SERVER_MODULE

En los puntos siguientes se desarrolla el proceso seguido para la adaptación del Wiimote_Server_Module, las modificaciones realizadas en el código original descargado de la página del autor y los resultados obtenidos.

4.1.1. CÓDIGO FUENTE UTILIZADO

Como se ha afirmado en puntos anteriores, es necesaria la librería Wiiuse mencionada para el funcionamiento del Wiimote_Server_Module.

El módulo está configurado para enviar únicamente los datos de los botones, no los de los demás sensores del *Wiimote* (que son de nuestro interés). Sin embargo, se pudo hallar una función en el fichero wiiuse.h llamada:

```
void wiiuse_motion_sensing (struct wiimote_t* wm, int status)
```

que tiene como parámetros un 'struct' con las distintas variables de los sensores.

Para conseguir habilitar esta función para que enviase los datos se vio que era necesario modificar funciones de varios ficheros de Wiimote_Server_Module y hasta de la propia librería subyacente, Wiiuse. Por tanto, se descartó el uso de esta vía para la transmisión de datos. Finalmente, se consideró la utilización de las funciones:

```
void calculate_orientation(struct accel_t* ac, struct vec3b_t* accel, struct orient_t* orient, int smooth);

void calculate_gforce(struct accel_t* ac, struct vec3b_t* accel, struct gforce_t* gforce);
```

Éstas se encuentran en dynamics.h (de la librería Wiiuse) y el procedimiento de adaptación a las necesidades específicas de lectura de datos se detalla en la siguiente subsección.

En definitiva, los ficheros modificados de Wiiuse y Wiimote_Server_Module fueron:

- WiimoteServerModule.cpp
- WiimoteServerThread.cpp
- WiimoteServerThread.h
- Dynamics.h

4.1.2. MODIFICACIONES REALIZADAS

Como parte de las funciones de inicialización del programa, se incluye la función:

```
void wiiuse_motion_sensing (struct wiimote_t* wm, int status)
```

en WiimoteServerThread.cpp. En WiimoteServerThread.h, se declararon las siguiente variables del 'struct' de la clase 'WiimoteServerThread':

```
typedef struct{
    float xg, yg, zg, x, y, z;  // Reserva de espacio de memoria para
                                //          los nuevos datos a transmitir
}
```

Para poder enviar la información por YARP, es necesario conocer el sistema que se ha usado. Lo más común, hablando de esta librería, es usar objetos de la clase "Bottle" para meter la información en los puertos, pero en el código escrito por Eric Sauser utiliza objetos de la clase "Vector" para enviar la información, que se envían por el puerto de YARP (clase "Port") de forma idéntica que un objeto de clase "Bottle":

Lin 109 de WiimoteServerThread.cpp :

```
Vector &outputVec = mOutputPort.prepare();
&ouputVec = mOutputPort.prepare ()
```

El resto de los puertos creados coinciden perfectamente en Wiiuse.h y WiimoteServerThread.h:

Puertos de entrada:

```
/WiimoteServer/moduleName/rpc
/* un sencillo Puerto para entrar en el modulo y seleccionar el modo de funcionamiento */
[emod]: event mode (default)
[smod]: stream mode
[quit]: quit the module (exit)
```

Puertos de salida:

```
/WiimoteServer/moduleName/output
/* Un vector de 11 posiciones que contiene todos los botones del mando en este orden*/
A, B, Up, Down, Left, Right, Minus, Plus, Home, One, Two
/*En modo event los valores son:*/
0 -> nada, 1 -> botón presionado, -1 -> botón no presionado
/*En modo streaming los valores del vector son*/
0 -> nothing, 1 -> button pressed
```

Se escribe al puerto de esta forma:

```
if(bEventMode)
    mOutputPort.writeStrict();  // Eventos, utilizado para datos de botones
else
    mOutputPort.write();  // Streaming, el utilizado para datos de sensores
```

Con estas modificaciones, y agregando el contenido de las nuevas variables de ambos sensores, el módulo debería enviar estos datos o al menos ser capaz de imprimirlos por pantalla antes de enviarlos por YARP (escribiendo sencillamente una línea con un 'printf' y las variables X,Y,Z, roll y pitch), pero no fue así. La canalización de los datos de los sensores hasta los espacios que se reservaron en memoria se desestimó debido a los factores explicados en la subsección 4.1.1, y se descartó la utilización de la función wiiuse_motion_sensing(struct wiimote_t* wm, int status).

Como alternativa se intentó conseguir el mismo objetivo, es decir, imprimir por pantalla los datos de las variables en tiempo de ejecución, pero esta vez usando otras funciones que se encuentran en dynamics.h:

```
void calculate_orientation(struct accel_t* ac, struct vec3b_t* accel, struct orient_t* orient, int smooth);
void calculate_gforce(struct accel_t* ac, struct vec3b_t* accel, struct gforce_t* gforce);
```

Para poder usarlas es necesario añadir el correspondiente 'include' en WiimoteServerThread.h:

```
lin 34    #include "dynamics.h"
```

y completar el struct con las nuevas variables que se van a usar (xg, yg, zg, x, y , z):

```
lin 54  typedef struct{
        char bA;
        char bB;
        char bUP;
        char bDOWN;
        char bLEFT;
        char bRIGHT;
        char bMINUS;
        char bPLUS;
        char bHOME;
        char bONE;
        char bTWO;
        char bNULL;
        float xg;
        float yg;
        float zg;
        float x;
        float y;
        float z; }
```

Para comprobar los resultados, con un 'printf' en WiimoteServerThread.cpp se muestran las variables por pantalla:

```
lin 168     printf("THESE ARE MY ROLL: %f and PITCH: %f. ",myRoll,myPitch);
lin 173     printf("THESE ARE MY GX: %f, GY: %f AND GZ:f.\n",myX,myY,myZ);
```

De nuevo los datos obtenidos son aleatorios e incoherentes, es decir, que no tienen en cuenta la posición del *Wiimote* y calculan erróneamente los parámetros como se muestra en la figura 4.1.

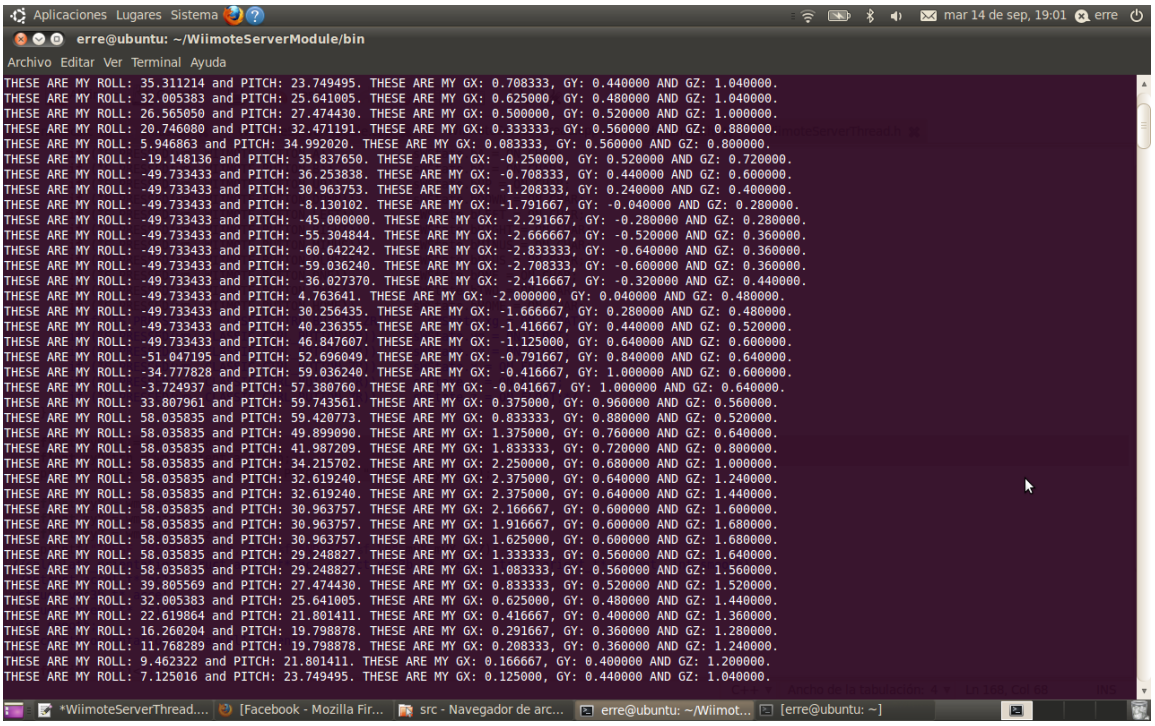


Figura 4.1: Datos obtenidos al usar Wiimote_Server_Module

En la siguiente sección (compilación y resultados) se detallan los pasos para compilar el código y ejecutarlo de forma que se obtiene el resultado de la imagen anterior.

4.1.3. COMPILACIÓN Y RESULTADOS

La compilación del código se realiza mediante el mecanismo CMake, por lo que se generará mediante este programa un archivo "Makefile" que posteriormente se procesará mediante "make".

El procedimiento es el siguiente:

- En primer lugar se crea un directorio build dentro de la carpeta del Wiimote_Server_Module que se pretende usar tecleando en un terminal:

```
mkdir build
cd build
cmake ..
make
```

- De esta forma se ha generado el archivo "Makefile" y en las sucesivas veces basta con borrar el contenido de la compilación anterior y volver a compilar, eso sí, dentro de la carpeta build que se ha creado:

```
rm –rf *
cmake ..
make
```

Una vez se ha compilado el código y se ha comprobado la ausencia de errores se ejecuta el programa creado tecleando en el terminal:

- Dirección de la carpeta indicada:

```
cd Wiimote_Server_Module
cd out
```

- Ejecutable:

```
./Wiimote_Server_Module
```

Al ejecutar el programa, se obtiene la lista de variables de la figura 4.1.

4.2. PRUEBA 2. CWIID Y WMGUI

En esta segunda prueba se utilizó la misma metodología que en la primera, desarrollando en primer lugar las modificaciones realizadas en el código para mostrar por pantalla los datos útiles que se necesitan como las modificaciones necesarias para compilar el código final y ejecutarlo. Finalmente se muestran los resultados obtenidos.

4.2.1. CÓDIGO FUENTE UTILIZADO

Una vez instalado el paquete completo de CWiid (que incluye Wmgui) que se ha descargado de la web y siguiendo los pasos explicados en la subsección 3.2.5 (software utilizado), fue necesario modificar ciertos elementos que se detallan en los siguientes puntos.

4.2.2. MODIFICACIONES REALIZADAS

Partiendo del código fuente de Wmgui en C, que contiene los ficheros main.c, interface.c e interface.h, se modificó main.c para poder imprimir por pantalla los datos numéricos que son enviados desde el *Wiimote* y poder trabajar con ellos.

```
printf ("\n ENVIADAS las variables del acc %2f, %2f, %2f, %2f, %2f,\n", a_x,a_y,a_z,roll, pitch);
```

Al funcionar todo perfectamente, mostrando por pantalla datos coherentes y acordes con el movimiento del mando inalámbrico según se muestra en la figura 5.1, no fue necesario realizar más modificaciones. La parte de comunicaciones se desarrollará en la sección 4.3 (compilación), donde se muestra que sí es necesario incluir varias líneas de código para poder compilar con CMake y usar la librería YARP.

4.3. COMPILACIÓN

A la hora de compilar el código que se ha modificado, dentro del archivo "readme" de CWiid se encuentran las directrices para dicho ejercicio.

El problema se encuentra en que el sistema utilizado por el desarrollador original es del tipo "automake" usando "./configure" que según se explicó en la sección 3.2.2, no es tan potente y multiplataforma como la herramienta CMake. Se optó por convertir este mecanismo al usado por todos los módulos desarrollados en este proyecto que se comunican mediante YARP y siguen las guías propuestas por RCGv03.

De nuevo fue necesario realizar modificaciones en el código de varios archivos para tal fin aunque en primer lugar, para comprobar que el código era compilable y ejecutable, se hizo la prueba según las siguientes indicaciones:

En un terminal se teclea:

```
./configure
```

En el caso de que no haya ningún error, quiere decir que compila correctamente, entonces en el mismo terminal se teclea:

```
make
```

Este comando genera un ejecutable automáticamente dentro de la carpeta de Wmgui que se puede ejecutar en cualquier momento para comprobar que el programa que se ha modificado funciona correctamente y acorde a las especificaciones que se tenían.

Para ejecutar el programa se teclea:

```
./Wmgui
```

En este momento, aparece la interfaz que se tenía anteriormente de CWiid pero con las modificaciones que se han realizado en main.c, ahora se pueden visualizar en el terminal las variables X, Y, Z, roll y pitch del *Wiimote* en este orden para su tratamiento posterior.

Una vez comprobado el correcto funcionamiento del programa, se procedió a la adaptación del mecanismo de compilación de automake a CMake.

Para compilar de esta forma se crea un directorio 'build' la primera vez, tecleando en el terminal:

```
mkdir build
cd build
cmake ..
make
```

y en las sucesivas se sustituye 'mkdir build' y 'cd build' por 'rm –rf *' para eliminar el contenido anterior.

4.3.1. MODIFICACIONES EN CÓDIGO

Para que funcione este proceso es necesario modificar ciertas líneas (cada vez que se modifique una línea de código hay que volver a compilar por el mecanismo CMake):

- En la carpeta de CMake que se generó anteriormente es necesario abrir el fichero CMakelists.txt e incluir en la línea 35:

```
FILE (GLOB folder_source ${MY_SRC_PATH}/*.c)
```

  Esta línea permite leer los ficheros de tipo ".c".

- Al compilar de nuevo, el error que se obtiene es el referido a la falta de la librería correspondiente a las variables de tipo gtk (vers. 2.0) que permiten el funcionamiento de la parte visual del programa.

  Mediante el mecanismo sudo apt-file search "poner paquete" se busca el paquete necesario, y una vez encontrado, se copia la ruta y se pega en la línea 11 de CMakelists.txt:

```
INCLUDE DIRECTORIES ( usr/include/gtk2.0 )
```

  A continuación se busca la librería correspondiente mediante el comando grep en el terminal tecleando: grep gtk-2.0 *

  Una vez encontrada, se escribe el nombre de la librería en la línea 57 de esta forma:

```
Target_link_libraries ({KEYWORD} gtk-x11-2.0)
```

  Al compilar se obtienen varios errores fruto de la inexistencia de los paquetes glib-2.0.h, cairo.h, pango-1.0.h y atk-1.0.h.

Con el comando sudo apt-file search nombre_paquete se buscan en el ordenador los paquetes necesarios y se incluye en CMakelists.txt la ruta de cada uno de ellos para establecer las dependencias:

```
INCLUDE DIRECTORIES ( usr/include/gtk2.0 /usr/include/glib-2.0 /usr/include/cairo /usr/pango-1-0 /usr/lib/gtk-2.0/include /usr/include/atk-1.0)
```

Se vuelve a compilar y se comprueba que no hay más errores.

- El siguiente paso consiste en buscar en el directorio lib del ordenador la librería de CWiid por la sencilla razón de que las variables de tipo gtk también tienen que ser reconocidos en esta librería. El mecanismo de búsqueda vuelve a ser grep cwiid.h *

  Una vez completada CWiid, ya se puede incluir en CMakelists.txt la librería CWiid en la línea

```
Target_link_libraries ({KEYWORD} gtk-x11-2.0 cwiid);
```

- Por último, en el main.c del programa CWiid se elimina en la línea 50 el comando #ifdef…. para que siempre utilice todo el código y también se incluye #include config.h porque se necesitarán algunas dependencias.

CAPÍTULO 5. PUESTA EN MARCHA Y VISUALIZACIÓN

Concluidos estos pasos, ya se ha conseguido compilar con el mecanismo CMake y en este momento es posible adaptar el programa a las necesidades de YARP.

5.1. APERTURA DE PUERTOS YARP

Al igual que el mecanismo CMake, es necesario volver a modificar ciertas líneas de código e incluir nuevas para conseguir adaptar YARP al programa y que compile sin problemas.

En este momento, y siguiendo las indicaciones de Robot Component Guidelines v0.3, se crea un directorio nuevo llamado Hmi_wiimote donde se incluyen todos los ficheros necesarios que antes se encontraban en el directorio Wmgui.

Volviendo a la parte de modificación de código, se realizan los siguientes cambios:

- En CMakelists.txt se incluye la línea FIND_PACKAGE (YARP)

Hay que tener en cuenta que se trabaja con una "variable de entorno", que viene a ser como una especie de variable del sistema operativo, por lo que al compilar va a dar un error.

Este error requiere asociar YARP_DIR al fichero donde se compila YARP, y para ello es necesario establecer una variable de entorno de la siguiente forma:

```
export YARP_DIR = ~/yarp-2.2.6/build
```

Al compilar de nuevo, se consigue eliminar este error, pero para que esta variable de entorno no desaparezca cada vez que se cierra el terminal hay que asociarla permanentemente en el archivo .bashrc de la forma:

```
export YARP_DIR= ~/yarp…./build
```

- Según el manual de YARP que se encuentra en http://eris.liralab.it/yarpdoc/index.html es necesario incluir en hmi_wiimote.cpp: #include "yarp/os/all"
- Una vez llegados a este punto, el problema es que la librería YARP está implementado casi íntegramente en C++, la librería CWiid con el que se está integrando está implementado completamente en C. Es importante remarcar que en un programa escrito en C++ es posible incluir partes en C, pero el compilador que se usará será de C++ con el inconveniente de ser más restrictivo (menos permisivo con código propenso a errores) que uno de C convencional.

Al compilar el código con YARP integrado se obtiene un error ya que dentro de la librería CWiid se está intentando enviar un valor de tipo int como uno de gtk_dialog_flag. Esto era admitido por un compilador en modo C, pero no en el modo de compilación C++ forzado por la dependencia YARP.

Como primera medida, se comentó la línea 489 de hmi_wiimote.cpp

```
dialog = gtk_message_dialog_new(parent, 0 , type, GTK_BUTTONS_OK, message);
```

El programa compila, pero se obtiene un fallo de segmentación al ejecutarlo. Esto se debe a que la línea comentada es fundamental para la inicialización de la parte visual del programa Wmgui, y por lo tanto causa un cierre de la aplicación cuando el programa accede a la ventana para mostrar las variables de forma gráfica unos instantes después de su ejecución.

Como segunda medida, y dado que el problema se tiene en el argumento 0 que hay en dicha línea, se cambia por otro propio de lenguaje C++, es decir, por un GTK_DIALOG_FLAG.

```
dialog = gtk_message_dialog_new(parent, GTK_DIALOG_FLAG, type, GTK_BUTTONS_OK, message);
```

Se vuelve a compilar y esta vez funciona y no se cierra de forma prematura.

También son necesarias las siguientes líneas para abrir un puerto de YARP:

```
Línea 215    BufferedPort mi Puerto;
Línea 222    Network yarp;
Línea 223    miPuerto.open ("/wiimote_info");
```

Por último, en el fichero hmi_wiimote.h son necesarias dos líneas de código:

```
Línea 36   #include "yarp/os/all.h"
           Using namespace yarp::os;
```

Una vez completados todos los pasos del tutorial se compila de nuevo y si no hay ningún error se ejecuta el programa.

Al funcionar todo correctamente, es la hora de preparar la información que se enviará mediante YARP una vez que se abran los puertos. Esta información se introduce en "hmi_wiimote.cpp" mediante el comando "Bottle" de la siguiente forma:

```
Línea 206     BufferedPort <Bottle> miPuerto;
Línea 213     Network yarp;
Línea 214     miPuerto.open("/wiimote_info");
Línea 1165    Bottle& miOutput = miPuerto.prepare ( )
              miOutput.addString ("accX");
              miOutput.addDouble (a_x);
              miOutput.addString ("accY");
              miOutput.addDouble (a_y);
              miOutput.addString ("accZ");
              miOutput.addDouble (a_z);
              miOutput.addString ("roll");
              miOutput.addDouble (roll);
              miOutput.addString ("pitch");
              miOutput.addDouble (pitch);
```

Se puede observar que la información que se agrega al paquete de datos a enviar por el puerto corresponde a la de las variables que se imprimían por pantalla en la subsección 4.2.2.

Una vez concluidos todos los pasos del tutorial, ya se puede empezar la aplicación con otras aplicaciones dispongan de posibilidades de interoperación con YARP.

5.2. VISUALIZACIÓN NUMÉRICA

La visualización numérica consiste sencillamente en imprimir por pantalla en un terminal los valores que da el mando *Wiimote Plus* de forma continua y ordenada.

Su puesta en marcha es también sencilla, basta con abrir un puerto YARP y conectar mediante el bluetooth con el mando. El proceso es el siguiente:

- En un primer terminal, teclear yarp server para inicializarlo.
- En un segundo terminal, dentro el directorio out/Linux-x86 del módulo Hmi_wiimote, se ejecuta de la forma ./hmi_wiimote

  La ruta completa es: ~/hmi_wiimote/out/Linux-x86/hmi_wiimote

  En este momento aparecerá el interfaz de Wmgui, en el cual se activará el mando *Wiimote Plus* siguiendo los pasos del punto 3.2.6 (Wmgui).

En el caso de tener cualquier problema, toda la documentación se encuentra en www.eris.liralab.it/yarp.com

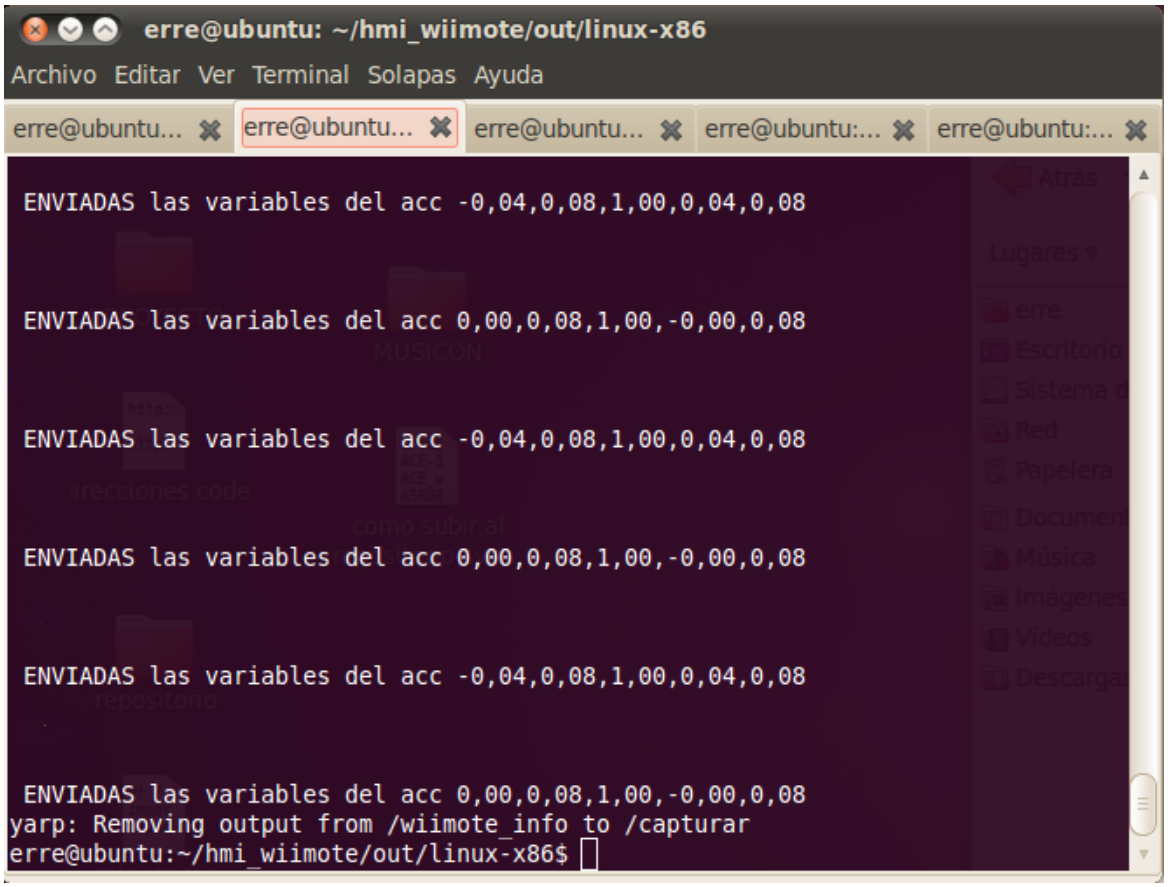


Figura 5.1: Visualización numérica de las variables del *Wiimote Plus*

5.3. VISUALIZACIÓN GRÁFICA

Según se ha comentado al inicio del documento, una vez concluida la parte de adaptación de software del módulo Hmi_wiimote propiamente dicho y de la comprobación de su buen funcionamiento mediante la visualización numérica, se va a mejorar su interpretación mediante una aplicación de visualización gráfica incluida en la librería YARP que se viene usando a lo largo de todo el documento.

El resultado son un conjunto de líneas que representan a las variables del mando *Wii Remote Plus* en este caso, pero que es aplicable a cualquier sensor de tipo Hmi_xsens.

El orden que se ha elegido para las variables es, en primer lugar, los tres posibles ejes de la aceleración medida por el *Wii Remote* convencional, y en segundo lugar las 2 variables que se han tomado del accesorio *Wii Motion Plus*, llamadas roll y pitch según se muestra en el siguiente diagrama.

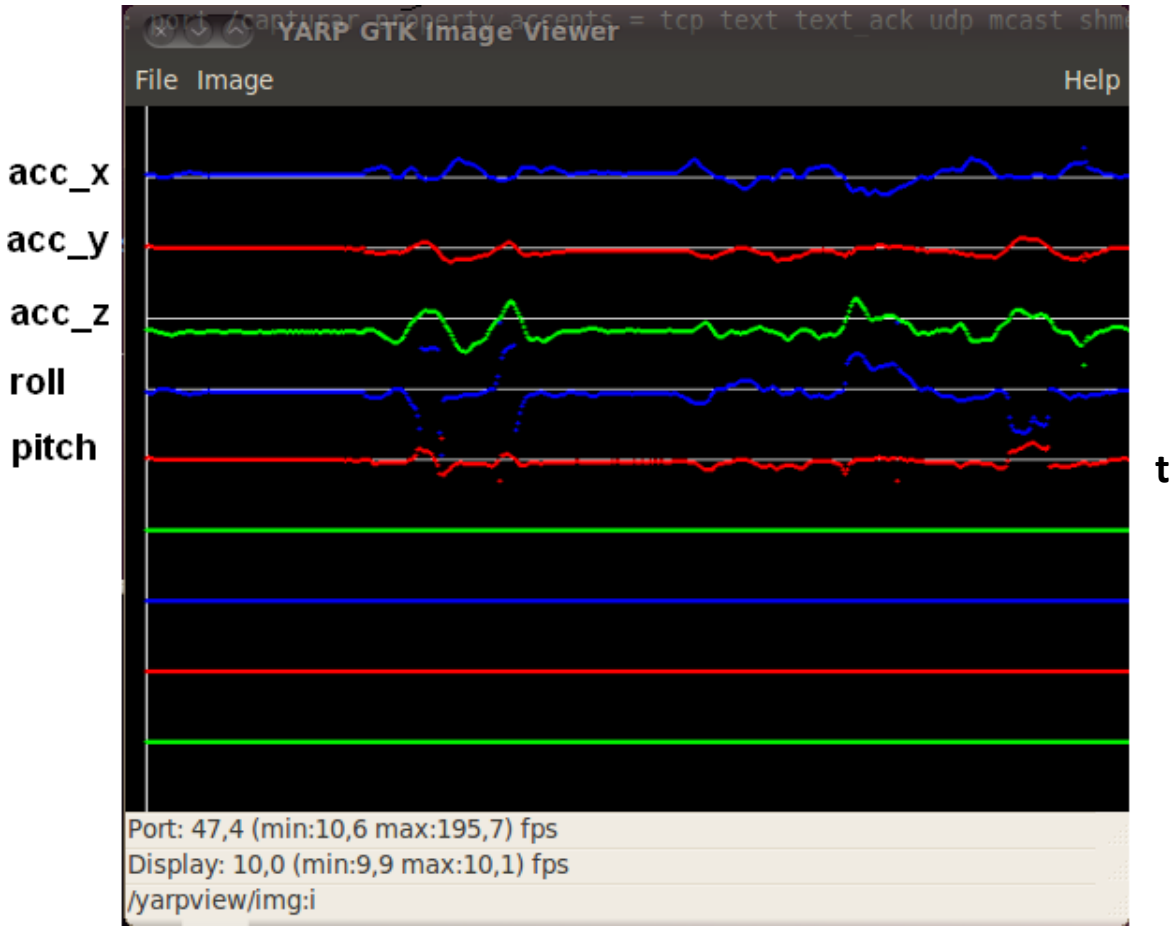


Figura 5.2: Visualización gráfica de las variables del *Wiimote Plus*

De esta manera se confirma que gracias a la estandarización de software que se ha conseguido siguiendo los Robot Comand Guidelines v0.3 es posible reutilizar los múltiples módulos que existen simplemente modificando ciertos detalles de calibración que se especifican a continuación en el apartado de modificaciones en código.

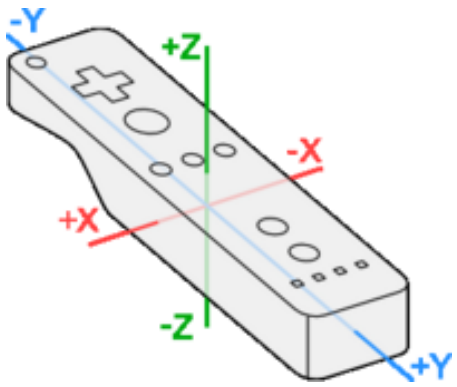


Figura 5.3: Sistema de ejes utilizado por el *Wiimote*

Para poner en marcha esta visualización, es necesario haber iniciado antes la numérica (punto 5.2), es decir, la gráfica depende de que le hayan llegado los datos al módulo anterior (Hmi_wiimote) para así abrir un segundo puerto YARP y enviar los datos al módulo "image" que los convierte en formato int para que los interprete correctamente la aplicación "yarpview".

Teniendo en cuenta que los tres primeros pasos son los mismos que para la visualización numérica, el proceso es el siguiente:

- En un primer terminal, teclear yarp server para inicializarlo.
- En un segundo terminal, dentro el directorio out/Linux-x86 del módulo Hmi_wiimote, se ejecuta de la forma ./hmi_wiimote

  La ruta completa es: ~/hmi_wiimote/out/Linux-x86/hmi_wiimote

  En este momento aparecerá el interfaz de Wmgui, en el cual se activará el mando Wii Remote siguiendo los pasos del punto 3.2.6 (Wmgui).
- En un tercer terminal, dentro del directorio de out/Linux-x86 del módulo Imagen creado por el compañero Daniel García, se ejecuta de la forma ./sns_xsens

  La ruta completa es: ~/imagen/out/Linux-x86/sns_xsens
- En un cuarto terminal, teclear yarpview para inicializar la aplicación de visualización de YARP.
- Por último, en un quinto terminal, ejecutar un pequeño archivo bash_script que se ha creado con el objeto de simplificar las conexiones entre los módulos y no tener que escribir la entrada y salida de los puertos cada vez que se quiere ejecutar el programa. El código se encuentra en el punto 5.3.1 referente a las modificaciones en código.

  Teclear en el terminal, ./conecta

En este momento ya se puede apreciar la variación de las aceleraciones en el monitor de Yarpview si se mueve el mando *Wiimote Plus*.

5.3.1. MODIFICACIONES EN CÓDIGO

Las modificaciones realizadas para que el módulo "imagen" represente de forma correcta las variables enviadas mediante YARP por el módulo Hmi_wiimote se reducen a un escalamiento de los valores de dichas variables en el código de hmi_wiimote.cpp.

Basta con introducir a partir de la línea 1260 los siguientes comandos:

```
Lin 1260    miOutput.addString("accX");
            miOutput.addDouble(a_x*6.0); //escalamos x6 para visualiz.
            miOutput.addString("accY");
            miOutput.addDouble(a_y*6.0);
            miOutput.addString("accZ");
            miOutput.addDouble(a_z*6.0);
            miOutput.addString("roll");
            miOutput.addDouble(roll*6.0);
            miOutput.addString("pitch");
            miOutput.addDouble(pitch*6.0);
```

Para mejorar la funcionalidad de ambos módulos cuando están trabajando simultáneamente, se procedió a crear el ya mencionado fichero "./conecta" que permite efectuar las conexiones de una forma mucho más rápida que manualmente en un terminal. Dicho fichero contiene las siguientes líneas:

```
#!/bin/sh
yarp connect /wiimote_info /capturar

yarp connect /imagen /yarpview/img:i
```

5.4. FUNCIONALIDAD DEL MÓDULO

El módulo Hmi_wiimote tenía como objetivo principal la transmisión de los datos del mando *Wiimote Plus* a través de YARP para visualizarlos posteriormente en una interfaz propia, pero aprovechando que está basado en las Robot Component Guidelines v0.3 como se ha indicado antes, tiene la virtud de ser reutilizable para su uso con multitud de dispositivos.

En nuestro caso se ha probado con el robot asistencial de cocina, conocido como ASIBOT, y mediante el *Wiimote Plus* se pueden manejar los 5 grados de libertad del robot.

Además, en vez de crear una interfaz de cero que podía resultar muy costoso en cuanto a tiempo se refiere, se ha reutilizado una creada por el compañero de trabajo Daniel García, demostrando así su compatibilidad absoluta y su gran funcionalidad.

El uso de este módulo es muy ventajoso ya que, simplemente con un ordenador, un dispositivo de bluetooth y el mando de la *Wii* se puede controlar cualquier dispositivo que esté basado en las mismas directrices de software y de forma muy económica.

Aparte de estas dos aplicaciones, que son las generales, existe una tercera que es la capacidad de visualizar los datos en un ordenador remoto gracias a la implementación de YARP. El proceso para ejecutar esta función es el siguiente:

- En primer lugar hay que descargar de la página de CWiid el software completo, incluído Wmgui e instalarlo en el ordenador (puntos 3.2.5 y 3.2.6).

- En segundo lugar se descarga el código correspondiente al módulo Hmi_wiimote del repositorio del departamento de robótica de la Universidad en el siguiente enlace y se instala en el ordenador (punto 4.3).

  http://roborepo.uc3m.es/svn/ASIBOTcoderepo/trunk/hmi_wiimote/

- Una vez instalados, se ejecuta el módulo Hmi_wiimote y se abre un puerto YARP siguiendo los pasos de la sección 5.2.

CAPÍTULO 6. CONCLUSIÓN Y FUTURAS AMPLIACIONES

En definitiva, ha quedado demostrado que los periféricos de la *Nintendo Wii* han abierto a la comunidad científica e investigadora una nueva vía para desarrollar proyectos y aplicaciones no sólo enfocados a personas discapacitadas o con enfermedades degenerativas, sino también como nuevas formas de control de dispositivos que hace unos años no era posible sin el desembolso de una buena cantidad de dinero en sensores que tuviesen una sensibilidad aceptable.

Concretamente, el dispositivo utilizado en este proyecto, el *Wiimote Plus* ha resultado ser una herramienta de control con un manejo sencillo y una gran cantidad de software libre desarrollado que ha facilitado enormemente su adaptación a las necesidades de transmisión de información. Tras las modificaciones realizadas se ha conseguido enviar dicha información mediante la herramienta YARP de una manera estandarizada que ha dotado al módulo de una enorme versatilidad como se demostró al ser capaz de conectarse mediante otra conexión YARP al módulo de imagen creado por Daniel García y más adelante moviendo el robot ASIBOT en cinco grados de libertad.

Gracias a esta arquitectura, el módulo Hmi_wiimote es reutilizable para cualquier aplicación que sea de la misma naturaleza y muy útil por su sencilla instalación y configuración en cualquier equipo según se desarrolla en la subsección 5.4.

En la actualidad, la gama de acelerómetros y sensores de medida de posición de uso doméstico está creciendo de forma exponencial como podemos comprobar con la gran aceptación que ha tenido la *Nintendo Wii*, el *iPhone® de Apple®*, otros smartphones con tecnologías similares, el nuevo periférico de *Sony* para la *Play Station 3*, el *MOVE* que se muestra en la figura 6.1.

Figura 6.1: Periférico MOVE para Playstation 3

El Move sustituye la barra de leds y el sensor de movimiento de la Wii por una cámara llamada Playstation Eye que, según sus creadores, detecta los ejes X e Y de la esfera con una precisión de milímetros, el eje Z de centímetros y todo ello con un tiempo de respuesta de 22 milisegundos. Puede resultar muy útil si se consigue procesar la información que recoge en un ordenador, pero habrá que esperar un tiempo y ver si tiene tanta aceptación como el Wiimote y se desarrolla software libre [12].

La otra superventas, Xbox 360 de Microsoft, está a punto de lanzar su homólogo, llamado en este caso Kinetc, [13] en la figura 6.2, que consiste en un dispositivo mucho más avanzado, con reconocimiento de voz, facial, de movimientos…

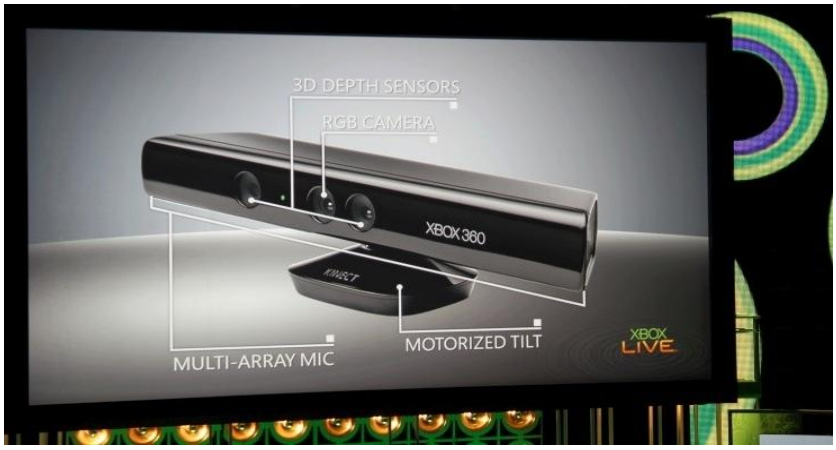


Figura 6.2: Sistema Kinetic de Microsoft

Además de estos nuevos dispositivos, *Nintendo* fabrica la *DS2*, una videoconsola portátil de la cual se ha hecho un análisis en el punto 2.5 y que puede ser otra vía para desarrollar proyectos, aparte de su potencial pedagógico y terapéutico, ya que dispone de software aplicado a mejorar la agilidad mental.

Otra innovación interesante en el contexto en que se está trabajando es una red de conexión entre los médicos y *Nintendo*, se llama "Check-up" y esta disponible en Japón a desde abril. Consiste en una guía entre el personal médico y el usuario de forma que se darán directrices saludables al jugador de *Wii Fit* y se controlará el progreso mediante el teléfono móvil. Ha sido desarrollado gracias a la colaboración de Hitachi®, NEC®, Panasonic® y una compañía de seguros.

En resumen, todos estos periféricos tan avanzados tecnológicamente hablando seguramente serán muy útiles a corto plazo y empezarán a usarse para aplicaciones como por ejemplo el caso del movimiento del Skybot que consiguió Juan González Gómez, compañero del departamento, en este caso usando el *Wiimote* adherido al brazo con cinta aislante a modo de brazo biónico.

Se puede comprobar el manejo del robot en el siguiente enlace, donde además se explica el proceso para su implementación:

http://www.iearobotics.com/proyectos/friki-apps/wii-skybot/wii-skybot.html

El uso de esta tecnología sólo está limitado por nuestra imaginación de modo que a diario se desarrollan nuevas aplicaciones cada vez más sofisticadas y útiles, como por ejemplo este mismo proyecto, que finalmente se ha concluido con el control de movimiento y posición del robot ASIBOT desarrollado por la Universidad Carlos III de Madrid, lo cual supone otra alternativa más al conjunto de soluciones para el colectivo de personas con algún grado de discapacidad en la parte superior del cuerpo y que a menudo se ven en apuros para realizar tareas cotidianas.

Figura 6.3: Doctor Mario. Imagen de la red “Check-up”

BIBLIOGRAFÍA

[1] Jardón Huete, Alberto (2006). Metodología de diseño de robots asistenciales. Aplicación al robot portátil ASIBOT. Departamento de Ingeniería de Sistemas y Automática, Universidad Carlos III de Madrid.

[2] González Víctores, J.Carlos (2010). Software Engineering Techniques Applied to Assitive Robotics: Guidelines & Tools. Departamento de Ingeniería de Sistemas y Automática, Universidad Carlos III de Madrid.

[3] Mary Saylor, Rebecca Setaro, Lynn Freeman, Barb Christensen (2010). Wii for Me. Communication Disorders Clinic, University of Central Florida.

[4] Aaron, S. (2009). Wii treatment for Parkinson's trialled. Medical College of Georgia.
www.nintendolife.com/news/2009/06/wii_treatment_for_parkinsons_trialled

[5] Redmond, Rebbeca (2009). Wii-Hab: Gaming's Contribution to Treating Parkinson's Disease. National Star College in Cheltenham.
http://www.triplepointpr.com/wii-hab-gaming%E2%80%99s-contribution-to-treating-parkinson%E2%80%99s-disease

[6] González Ingelmo, Elena (2009). Wii terapia. Centro de Referencia Estatal de Salamanca.
http://www.imserso.es/crealzheimer_01/terapias_no_farmacologicas/wii_terapia/index.htm

[7] Mantilla-Gomez, Fernando, Palacio-González, Angel (2009). An accessibility framework based on WiiMote. International Conference on Applied Informatics And Communications, Oviedo.

Muñoz, A.M., Duboy, M.A.V.(2009). A low-cost multimodal device for web accessibility. TSIC, Universidad Politécnica de Madrid.

[8] Pastor Herrán, Carlos (2010). Prototipado rápido de aplicaciones Opensource para rehabilitación usando dispositivos inalámbricos. Departamento de Ingeniería de Sistemas y Automática, Universidad Carlos III de Madrid.

[9] Palleja, T., Rubión, E., Teixido, M., Tresanchez, M., del Viso, A.F., Rebate, C. and Palacín, J. (2008). Using the optical flow to implement a relative virtual mouse controlled by head movements. Journal of Universal Computer Science. University of LLeida.

[10] Placer, Victor (2009). Migración de la plataforma a bordo del robot asistencial ASIBOT. Universidad Carlos III de Madrid.

[11] Victores, J.G. and Jardón, A. and Bonsignorio, F. and Stoelen, M.F. and Balaguer, C. (2010)Usability of Assistive Robotic Systems: Methodology and Application. Workshop on The Role of Experiments in Robotic Research at Icra, Anchorage, Alaska. Universidad Carlos III de Madrid.

[12] Página web Playstation Move. http://us.playstation.com/ps3/playstation-move/

[13] Página web Xbox Kinect. http://www.xbox.com/es-ES/kinect/

Página web de contenido YARP y tutorial.

http://eris.liralab.it/yarpdoc/index.html

Página de iCub y documentación de Wiimote_Server_Module.

http://eris.liralab.it/iCub/main/dox/html/group__icub____lasaImitation__WiimoteServerModule.html

Página web de Wiiuse

http://www.wiiuse.net/

Página web de CWiid y Wmgui

http://abstrakraft.org/cwiid/

http://abstrakraft.org/cwiid/browser/wmgui/interface.c

Página web de Robotics Lab de la Universidad Carlos III de Madrid (RCGv03).

http://robots.uc3m.es/w/index.php/Main_Page

Repositorio ASIBOT.

http://roborepo.uc3m.es/svn/ASIBOTcoderepo/

ANEXO A: MODIFICACIONES DE CÓDIGO

MODIFICACIONES DE CÓDIGO FUENTE: PRIMERA PRUEBA

Modificaciones en Wiimote_Server_Module.cpp:

Unmodified Added Removed

Personal/jgvictores/PFCs/rRodrigo/2010-11-17/WiimoteServerModule.cpp Tabular | Unified

r751	r752	
28	28	int main(int argc, char *argv[])
29	29	{
	30	printf("HELLO\n");
30	31	Network yarp;
31	32	if(!yarp.checkNetwork())
...	...	
99	100	wiiuse_rumble(WiimoteServerThread::sWiimotes[0], 0);
100	101	wiiuse_set_leds(WiimoteServerThread::sWiimotes[0], WIIMOTE_LED_2|WIIMOTE_LED_3);
	102	wiiuse_motion_sensing(WiimoteServerThread::sWiimotes[0], 1); // en dynamics.c los argumentos de la funcion son punteros
101	103	
102	104	char portName[255];
...	...	
148	150	index++;
149	151	break;
	152	//NOS INTERESA ESTE CASO PARA EL ACELEROMETRO. en thread.cpp bEventMode tambien false
150	153	case VOCAB4('s','m','o','d'):
151	154	mThread->SetEventMode(false);

Modificaciones en Wiimote_Server_Thread.cpp:

Unmodified Added Removed

Personal/jgvictores/PFCs/rRodrigo/2010-11-17/WiimoteServerThread.cpp Tabular | Unified

r751	r752	
45	45	bool WiimoteServerThread::threadInit()
46	46	{
	47	
47	48	char portName[256];
48	49	snprintf(portName,256,"/%s/output",mBaseName);
...	...	
105	106	
106	107	// Write data to output port
	108	// AQUI MODIFICO EL FOR PARA PREPARARLO PARA LAS NUEVAS VARIABLES DEL ACELEROMETRO 11+6
107	109	Vector &outputVec = mOutputPort.prepare();
108		outputVec.resize(11);
109		for(int i=0;i<11;i++){
	110	outputVec.resize(17);
	111	for(int i=0;i<17;i++){
110	112	char c;
111	113	if(bEventMode) c = ((char*)&sWEventState)[i];
	114	// TENGO QUE METER AKI TAMBIEN OTRO CONTADOR DE 7 PARA LA VARIABLE Y????????? $$$$$$
112	115	else c = ((char*)&sWState)[i];
113	116	if(c=='x')
...	...	
134	137	
135	138	#define ON_CHAR 'x'
	139	#define ON_FLOAT 'y'
136	140	
137	141	void WiimoteServerThread::handle_event(struct wiimote_t* wm) {
...	...	

```
...  ...
141  145
142  146     if (wm->btns) {
143              if (IS_PRESSED(wm, WIIMOTE_BUTTON_A))     sWState.bA = ON_CHAR;
144              if (IS_PRESSED(wm, WIIMOTE_BUTTON_B))     sWState.bB = ON_CHAR;
145              if (IS_PRESSED(wm, WIIMOTE_BUTTON_UP))    sWState.bUP = ON_CHAR;
146              if (IS_PRESSED(wm, WIIMOTE_BUTTON_DOWN))  sWState.bDOWN = ON_CHAR;
147              if (IS_PRESSED(wm, WIIMOTE_BUTTON_LEFT))  sWState.bLEFT = ON_CHAR;
148              if (IS_PRESSED(wm, WIIMOTE_BUTTON_RIGHT)) sWState.bRIGHT = ON_CHAR;
149              if (IS_PRESSED(wm, WIIMOTE_BUTTON_MINUS)) sWState.bMINUS = ON_CHAR;
150              if (IS_PRESSED(wm, WIIMOTE_BUTTON_PLUS))  sWState.bPLUS = ON_CHAR;
151              if (IS_PRESSED(wm, WIIMOTE_BUTTON_ONE))   sWState.bONE = ON_CHAR;
152              if (IS_PRESSED(wm, WIIMOTE_BUTTON_TWO))   sWState.bTWO = ON_CHAR;
153              if (IS_PRESSED(wm, WIIMOTE_BUTTON_HOME))  sWState.bHOME = ON_CHAR;
154          }
     147          if (IS_PRESSED(wm, WIIMOTE_BUTTON_A))           sWState.bA = ON_CHAR;
     148          if (IS_PRESSED(wm, WIIMOTE_BUTTON_B))           sWState.bB = ON_CHAR;
     149          if (IS_PRESSED(wm, WIIMOTE_BUTTON_UP))          sWState.bUP = ON_CHAR;
     150          if (IS_PRESSED(wm, WIIMOTE_BUTTON_DOWN))        sWState.bDOWN = ON_CHAR;
     151          if (IS_PRESSED(wm, WIIMOTE_BUTTON_LEFT))        sWState.bLEFT = ON_CHAR;
     152          if (IS_PRESSED(wm, WIIMOTE_BUTTON_RIGHT))       sWState.bRIGHT = ON_CHAR;
     153          if (IS_PRESSED(wm, WIIMOTE_BUTTON_MINUS))       sWState.bMINUS = ON_CHAR;
     154          if (IS_PRESSED(wm, WIIMOTE_BUTTON_PLUS))        sWState.bPLUS = ON_CHAR;
     155          if (IS_PRESSED(wm, WIIMOTE_BUTTON_ONE))         sWState.bONE = ON_CHAR;
     156          if (IS_PRESSED(wm, WIIMOTE_BUTTON_TWO))         sWState.bTWO = ON_CHAR;
     157          if (IS_PRESSED(wm, WIIMOTE_BUTTON_HOME))        sWState.bHOME = ON_CHAR;
     158  /*        if (IS_PRESSED(wm, CLASSIC_CTRL_BUTTON_ZR))     sWState.xg = ON_FLOAT;
     159          if (IS_PRESSED(wm, CLASSIC_CTRL_BUTTON_ZR))     sWState.yg = ON_FLOAT;
     160          if (IS_PRESSED(wm, CLASSIC_CTRL_BUTTON_ZR))     sWState.zg = ON_FLOAT;
     161          if (IS_PRESSED(wm, CLASSIC_CTRL_BUTTON_ZR))     sWState.x = ON_FLOAT;
     162          if (IS_PRESSED(wm, CLASSIC_CTRL_BUTTON_ZR))     sWState.y = ON_FLOAT;
     163          if (IS_PRESSED(wm, CLASSIC_CTRL_BUTTON_ZR))     sWState.z = ON_FLOAT;*/
     164      }
     165
     166      float myRoll = wm->orient.a_roll;
     167      float myPitch = wm->orient.a_pitch;
     168      printf("THESE ARE MY ROLL: %f and PITCH: %f. ",myRoll,myPitch);
     169
     170      float myX = wm->gforce.x;
     171      float myY = wm->gforce.y;
     172      float myZ = wm->gforce.z;
     173      printf("THESE ARE MY GX: %f, GY: %f AND GZ: %f.\n",myX,myY,myZ);
     174  //calculate_orientation(struct accel_t* ac, struct vec3b_t* accel, struct orient_t* orient, int
          smooth);
     175   /*struct accel_t* ac1;
     176   struct vec3b_t* accel1;
     177   struct orient_t* orient1;
     178   int smooth;
     179   calculate_orientation(ac1,accel1,orient1,1);
     180  */
155  181      for(int i=0;i<11;i++){
156  182          if(((char*)&sWState)[i] != ((char*)&sWPrevState)[i]){
...  ...
160  186                  ((char*)&sWEventState)[i] = '.';
161  187              sGotEvent = true;
     188      //nuevo contador para acelerometro
     189
     190  /*        for(int i=0;i<7;i++){
     191              if(((char*)&sWState)[i] != ((char*)&sWPrevState)[i]){
     192                  if(((char*)&sWState)[i]==ON_CHAR)
     193                      ((char*)&sWEventState)[i] = 'y';
     194                  else
     195                      ((char*)&sWEventState)[i] = ':';
     196                  sGotEvent = true;
     197      */
162  198          }
163  199      }
```

MODIFICACIONES DE CÓDIGO FUENTE: SEGUNDA PRUEBA

Modificaciones en main.c, renombrado como hmi_wiimote.cpp:

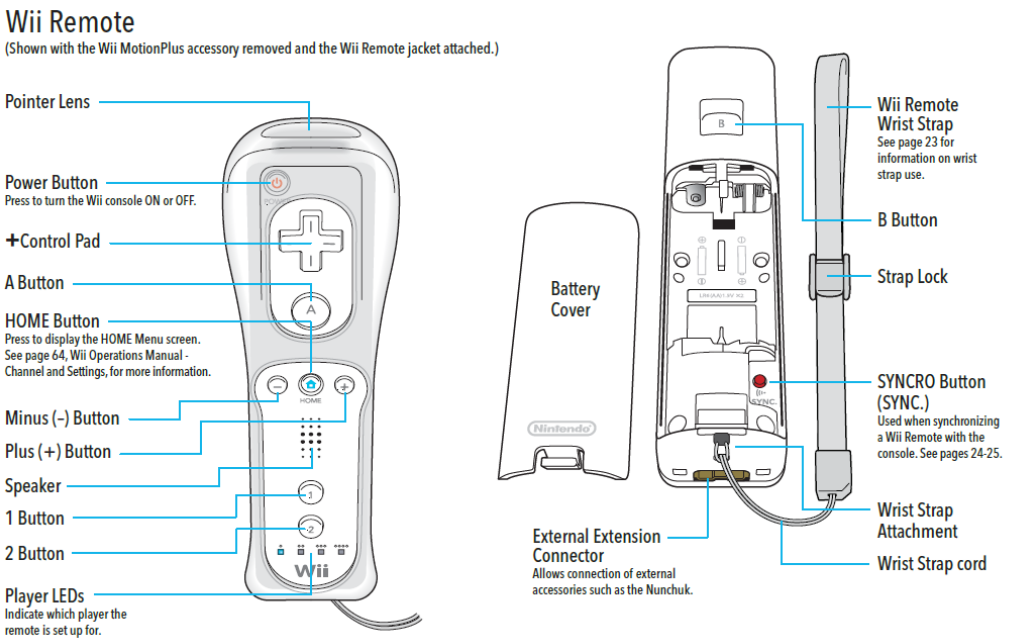
Personal/jgvictores/PFCs/rRodrigo/2010-11-17/main.c — Tabular | Unified

r755	r756	
48	48	*/
49	49	
50		#ifdef HAVE_CONFIG_H
51	50	#include "config.h"
52		#endif
53	51	
54	52	#define APP_NAME "CWiid wmgui"
...	...	
75	73	#include <bluetooth/bluetooth.h>
76	74	#include "cwiid.h"
	75	
	76	#include "hmi_wiimote.h"
77	77	
78	78	#define PI 3.14159265358979323
...	...	
204	204	*/
205	205	
	206	BufferedPort<Bottle> miPuerto;
	207	
206	208	int main (int argc, char *argv[])
207	209	{
208	210	int c;
209	211	char *str_addr;
	212	
	213	Network yarp;
	214	miPuerto.open("/wiimote_info");
210	215	
211	216	gtk_set_locale ();
...	...	
451	456	{
452	457	GtkWidget *dialog;
453		
454		dialog = gtk_message_dialog_new(parent, 0, type, GTK_BUTTONS_OK, message);
	458	// ELIMINO EL 0 DE LOS ARGUMENTOS
	459	dialog = gtk_message_dialog_new(parent, GTK_DIALOG_MODAL, type, GTK_BUTTONS_OK, message);
455	460	gtk_dialog_run(GTK_DIALOG(dialog));
456	461	gtk_widget_destroy(dialog);
...	...	
1024	1029	char *ext_str;
1025	1030	static enum cwiid_ext_type ext_type = CWIID_EXT_NONE;
	1031	printf("\nhola\n");
1026	1032	
1027	1033	gdk_threads_enter();
...	...	
1110	1116	gtk_widget_modify_bg(ev2, GTK_STATE_NORMAL,
1111	1117	(mesg->buttons & CWIID_BTN_2) ? &btn_on : &btn_off);
	1118	//printf ("\n aqui tengo los botones %f,%f,%f,%f,%f \n",a_x,a_y,a_z,roll,pitch);
1112	1119	}
1113	1120	
...	...	
1155	1162	g_snprintf(str, LBLVAL_LEN, "%.2f", pitch);
1156	1163	gtk_label_set_text(GTK_LABEL(lblPitchVal), str);

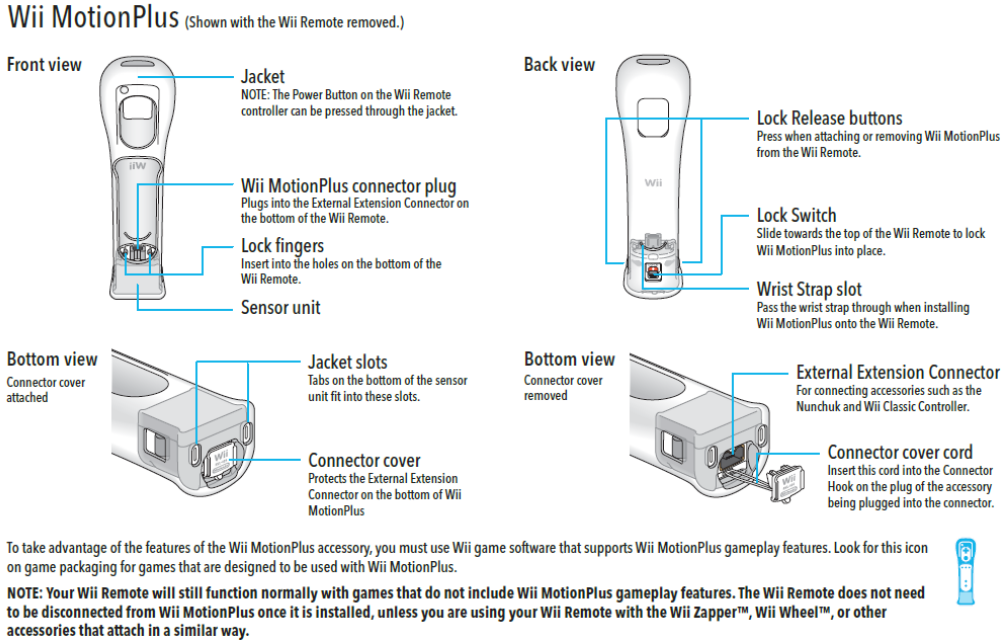
```
                Bottle& miOutput = miPuerto.prepare();
                miOutput.clear();
                miOutput.addString("accX");
                miOutput.addDouble(a_x*6.0); // escalamos x6 para el visualizador
                miOutput.addString("accY");
                miOutput.addDouble(a_y*6.0);
                miOutput.addString("accZ");
                miOutput.addDouble(a_z*6.0);
                miOutput.addString("roll");
                miOutput.addDouble(roll*6.0);
                miOutput.addString("pitch");
                miOutput.addDouble(pitch*6.0);

                miPuerto.write(true);

                printf ("\n ENVIADAS las variables del acc %.2f,%.2f,%.2f,%.2f,%.2f
\n",a_x,a_y,a_z,roll,pitch);

        }
}
```

ANEXO B: HOJAS DE CARACTERÍSTICAS

DATASHEET: WII REMOTE



DATASHEET: WII MOTION PLUS



To take advantage of the features of the Wii MotionPlus accessory, you must use Wii game software that supports Wii MotionPlus gameplay features. Look for this icon on game packaging for games that are designed to be used with Wii MotionPlus.

NOTE: Your Wii Remote will still function normally with games that do not include Wii MotionPlus gameplay features. The Wii Remote does not need to be disconnected from Wii MotionPlus once it is installed, unless you are using your Wii Remote with the Wii Zapper™, Wii Wheel™, or other accessories that attach in a similar way.

DATASHEET: ZOOM 4310

zoom®

Bluetooth®
Wireless Technology

Adaptors

Model 4310 USB Adaptor
Model 4312 PC Card Adaptor

The Zoom Bluetooth Adaptors plugs into a personal computer to provide a wireless connection to another Bluetooth-enabled device, such as a modem, mobile phone, headset, computer, PDA or peripheral. Zoom Bluetooth adaptors are Class 1 Bluetooth devices, which allows them to provide a range of up to 100 meters (330 feet)*, the longest range provided under the Bluetooth standard. Zoom's Class 1 Bluetooth device will communicate with all other Bluetooth devices, including Class 1, Class 2 and Class 3 devices. When connecting to Class 2 and Class 3 devices, Zoom Bluetooth adaptors are limited to the significantly shorter range of the less powerful Bluetooth devices on the other end of the transmission.

Zoom Bluetooth adaptors support 723,000 data throughput, the maximum allowed by the Bluetooth standard.

The Model 4310 is a USB adaptor which plugs into the USB port of a Macintosh with OS X or greater or a Windows XP, Me, 2000, or 98SE computer. The Model 4310 includes an external antenna to increase range.

The Model 4312 is a PC Card which plugs into the PC Card slot of a Windows XP, Me, 2000, or 98SE computer. An external antenna can be extended from the body of the PC Card to increase range.

Zoom Bluetooth adaptors come complete with a software installation wizard which makes setup of the Bluetooth protocol stack fast and easy.

The Bluetooth Piconet protocol supports concurrent connections with up to seven other Bluetooth devices. Security is ensured with the implementation of a Bluetooth security key and data encryption.Each Zoom Bluetooth adaptor contains a unique link key that prevents unauthorized access to the attached computer. The key is transmitted in a secure environment to prevent interception. Unlike an 802.11 b/g wireless device, Bluetooth wireless links are not susceptible to eavesdropping and false authentication.

Zoom Bluetooth adaptors carry a two year warranty. Documentation is provided in English, Spanish and Portuguese.

Features:

- Class 1 Bluetooth Device with a maximum range of 100 meters (330 feet) *
- Communicates with Class 1, 2, and 3 Bluetooth devices
- Includes setup wizard for fast and easy software installation
- Provides secure wireless communications at data rates up to 723K bps
- Bluetooth V1.1 compliant
- Two year warranty
- Documentation provided in English, Spanish and Portuguese

Zoom Bluetooth Wireless Technology Adaptors

Model 4310
Model 4312

Hardware Specifications for both Model 4310 and 4312

- Compliant with Bluetooth Version 1.1
- Class 1 Bluetooth device with a range of up to 100 meters (330 feet)
- Connects concurrently with up to seven other Bluetooth devices
- Operates in the license-free 2.4 GHz to 2.4835 GHz frequency band
- 723K bps data rate
- Two-color LED status indicator: USB wake up, transmitting
- FCC and CE certification

Model 4310 Hardware Specifications

- Full-speed USB Version 1.1 interface
- Pivoting single sleeve dipole antenna, 60.2 mm long (2.37 inches)
- Dimensions: 72.2 mm long X 29.8 mm wide X 8.5 mm high (2.84" X 1.17" X .33"), width includes the antenna
- Power consumption Typical: 110mA/5V; Maximum transmit: 200mA/5V; Maximum receive:80mA/5V; Idle: 26mA/5V

Model 4312 Hardware Specifications

- PC Card (PCMCIA) Type II fits Type II, III or Toshiba 16mm slot
- Dipole antenna can be extended from the body of the PC card, 63.5 mm long antenna (2.5 inches)
- Power consumption Typical: 110mA/5V; Maximum transmit: 200mA/5V; Maximum receive: 80mA/5V; Idle: 26mA/5V

Software Specifications for both Model 4310 and 4312

- Bluetooth V1.1 protocol stack
- The following Bluetooth networking profiles are provided: General Access Profile (GAP), Service Discovery Application Profile (SDAP), Serial Port Profile (SPP), Dial-up Networking Profile (DUN), Fax Profile, Local Area Network Access Profile (LAP), Generic Object Exchange Profile (GOEP), Object Push Profile (OPP), File Transfer Profile (FTP), Headset Profile, Personal Area Network Profile (PAN)
- Security software using encryption, authentication, paring
- Drivers supported by Model 4310 USB: Windows 98SE, Me, 2000, XP, Macintosh OS 10.2 and higher
- Drivers supported by Model 4312 PC Card: Windows 98SE, Me, 2000, XP

Package contents:

Model 4310 USB Adaptor

- USB adaptor with Bluetooth Class 1 wireless technology Installation CD for Windows XP, ME, 2000 and 98SE operating systems (Macintosh OS 10.2 and later includes Bluetooth networking protocols compatible with the Model 4310)
- Quick Start Guide
- Two-year warranty

Model 4312 PC Card

- PC Card with Bluetooth Class 1 wireless technology Installation CD for Windows XP, ME, 2000 and 98SE operating systems
- Quick Start Guide
- Two-year warranty

* Range may be less due to interference, physical obstructions or other conditions.

2933431O/12

zoom

International Headquarters
Zoom Telephonics, Inc.
207 South Street
Boston, MA 02111
USA
Tel: 617 423-1072
Fax: 617 423-3923

Nasdaq: ZOOM

email: sales@zoom.com

Website: www.zoom.com

European Sales/Support
Zoom/Hayes
430 Frimley Business Park
Frimley, Camberley
Surrey, GU16 7SY UK
Tel: +44 (0) 1276 704400
Fax +44 (0) 1276 704500

PRESUPUESTO

PRESUPUESTO HARDWARE				
CODIGO	**UDS.**	**DETALLE**	**PRECIO UD.**	**TOTAL**
01.1	1	**Mando *Wii Motion Plus* de *Nintendo:*** mando inalámbrico con accesorio Motion Plus que incorpora el sensor giroscópico MEMS MPU 3000.	55.95 €	55.95 €
01.2	1	**Dispositivo USB Bluetooth Zoom 4310:** adaptador de bluetooth para ordenador personal compatible con versión 1.1 en adelante.	28.00 €	28.00 €
PRECIO TOTAL				**83.95 €**